
أكثر من 90 وصفة عملية لإدارة فعالة، وتصميم الحلول باستخدام PostgreSQL
كتاب وصفات
POSTGRESQL
إجابات سريعة لأكثر المشاكل شيوعا

بوستجريسكل كتاب الوصفات

أكثر من ٩٠ وصفة عملية لإدارة فعالة، وتصميم الحلول باستخدام PostgreSQL

تأليف:

تشيتيج تشوهان

ترجمة:

راضية بن مني

مراجعة:

فهد بن عامر السعيدي **زايد بن عامر السعيدي**

بدعم من:

وادي التقنية

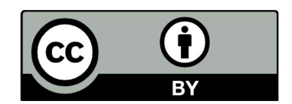
www.itwadi.com

نسخة ١.٠

١٤٣٨هـ - ٢٠١٧م

رخصة الكتاب

أصل هذا العمل ترجمة متصرفة لكتاب PostgreSQL Cookbook ، تأليف تشيتيج تشوهان، قام بالترجمة العربية موقع وادي التقنية، وبدعم سخي من المهندس/ زايد بن عامر السعيدي، وإشراف عام ومراجعة فهد بن عامر السعيدي.





عن المؤلف

يعمل تشتيج تشوهان حاليا كمسؤول قاعدة بيانات رفيع المستوى في MNC لتكنولوجيا المعلومات والتي مقرها في شانديغار. لديه أكثر من ١٠ سنوات من الخبرة في مجال قاعدة البيانات وإدارة النظام، مع التخصص في عناقيد MySQL ، وبوستجريسكل، و Greenplum، و Informix DB2، و Sybase, SQL Server2008، وأوراكل. وهو خبير بارز في مجال أمن قاعدة البيانات، مع خبرة في منتجات أمن قاعدة البيانات مثل IBM InfoSphere Guardium، وقاعدة بيانات أوراكل فولت، و Imperva.

عن وادي التقنية

وادي التقنية موقع تقني عربي يُعنى بتتبع أخبار البرمجيات الحرة والمواد التعليمية المتعلقة بها، يكتب فيه عدد من المتطوعين المهتمين بالبرمجيات الحرة والتقنية بشكل عام، يهتم وادي التقنية بمواضيع مثل أنظمة التشغيل الحاسوبية و الهاتفية ، لغات البرمجة ، مكتبات البرمجية ، تقنيات الويب ، أخبار شركات البرمجة الكبرى ، المصادر المفتوحة ، العتاد ، و أجهزة الحاسوب والهواتف.

وادي التقنية قام بدعم كتابة العديد من الكتب التقنية في مجال البرمجيات الحرة ومفتوحة المصدر، وتوفيرها مجانا للمستخدم التقني العربي، من أهم الكتب التي دعمها وادي التقنية:

تعلم جافا سكربت، دفتر مدير دبيان، سطر أوامر ليئكس، انطلق في انكسكيب، تعرف على البرمجيات الحرة

لزيارة وادي التقنية وتقديم الدعم له من هنا:

http://itwadi.com/

مقدمة لا بد لها:

الألفة من أقوى دوافع قبول الأشياء التي لم تخطر على البال، وحين تحل تجد أن التغيير إلى غيرها أمر في غاية المشقة و شيء كبير يحتاج إلى الشجاعة، وكثير من الأحيان إن لم يكن أغلبه يشترى الناس الألفة بالمال، فهم ينشدون السلام الفكري ومن لا يحب السلام؟ هكذا يخيل لهم، وعلى هذا الوتر تجد الشركات و المشاهير يبذلون المال و كثيرا من التفاهات ليبقوا في عقول الناس صورة ذاك السلام الذهني.

ومن أخطر أو أفضل (حسب موقفك من الأمر) أن تبدأ ترسيخ هذه الصور حين يكون الإنسان لا يملك معرفة عن الأمر أو أنه لم يطور ملكة النقد في مراحل مراهقته و سني فتوة شبابه؛ لذا ترى أن الشركات تتسابق في البذل الكثير للطلاب في المدارس والجامعات، و ربما بتشجيع من إدارة تلك المؤسسات.

وشركات التقنية لم تكن بدعا من الأمر، وربما كانت مهمتها أسهل في الدول الفقيرة ماليا و ثقافيا ، لذا تجدها تارة توزع برامجها بالمجان و تارة بأسعار مخصصة للطلاب، وتجد المسابقات الكثيرة التي ترعاها ، والأخطر أنها تدخل تعليم برامجها في المناهج الدراسية لتحوز قصبة السبق لعقول الطلاب الذين سيكونون صناع القرار في المستقبل.

على مر السنين كان هناك بديل عن ما هو مشهور، و ليس كل ما هو مشهور هو الأفضل ، و مسألة التفضيل من مسائل الأذواق و التي لا تستقيم إلا حين تقيم بحقائق لا تحمل تفضيلا بحد ذاتها، وعلى هذا المنوال اشتهرت في بلادنا تفضيل منتجات مايكروسوفت ليس لأنها الأفضل بل كان دور الدعاية وقربها من صناع القرار ذاع صيتها فأصبحت تدرس في المدارس والجامعات وكأنها لا يوجد غيرها خصوصا في بيئات العمل التجاري من أمثال قواعد البيانات و مخدمات البريد الإلكتروني وغيرها.

والأمر الذي يعنينا في هذا الكتاب هو قواعد البيانات ، وهي برامج حاسوبية تخزن و تسترجع و تبحث بشكل منظم في المعلومات التي تهم مستخدم النظام الحاسوبي، و هذا فن قديم قدم الحاسوب ، و كان هذا نوع من التطبيقات أحد دوافع لتطوير الحاسوب، ولأصالة هذا المجال تجد في السوق الكثير من برامج قواعد البيانات، الصغير و الكبير، المغلق و مفتوح المصدر ، المجاني و مدفوع الثمن ، ومنها ما ضرب لها ثمن باهظ و أخرى زهيدة.

ومن أشهر هذه التطبيقات بوستجريسكل PostgreSQL، وستجد شرحا وافيا عنها في هذا الكتاب، والكتاب يفتح الباب و لا يستوعب هذه البرنامج، مرتب على أبواب تعطيك مداخل لتقارن ما تعرفه من قبل، ونحن نشجعك على تعرف على قاعدة البيانات هذه ، و أن لا تجعل ما تعرفه من قبل عائق عن تقييم قدرات قاعدة بيانات بوستجريسكل.

نحن في وادي التقنية نضع هذا الكتاب ولا نطمع في شيء غير نشر العلم، و قد بذلنا مالنا و جهدنا و وقتنا في سبيل ذلك، وقد عزمنا على الأمر لا نألوا على المخذلين من صناع القرار أو إعراض الناس عن القراءة، إنا نؤمن بأن نشر العلم بلسان أهل البلد يفتح فرصة لطامحين لتغيير وتحسين من قدراتهم و حظوظهم من العلم.

وفي الختام ندعو كل من لديه مشروع لتأليف كتاب أو قدرة على الترجمة الجيدة باللغة العربية أن يتواصل معنا، لعل الله أن يوفقنا على إخراج المزيد من الكتب المفيدة في مجال تقنية المعلومات.

نسأل الله القبول و التوفيق.

زايد بن عامر السعيدي

سلطنة عمان

المحتويات

مدخل

«PostgreSQL» هي قاعدة بيانات تتوفر على مجموعة واسعة من المنصات وهي واحدة من قواعد البيانات المفتوحة المصدر الأكثر شعبية المنتشرة في بيئات الإنتاج في جميع أنحاء العالم.

كما أنها واحدة من قواعد البيانات الأكثر تطورا، وتحوي مجموعة واسعة من المميزات التي تنافس حتى العديد من قواعد البيانات المملوكة التجارية. هذا الكتاب يقدم لك نظرة ثاقبة ومعمقة في مختلف هذه الميزات وتطبيقاتها على «بوستجريسكل». فهو يهدف إلى أن يكون دليلاً عمليا لمديري قواعد البيانات ومطوري البرامج على حد سواء، مع تقديم الحلول المتعلقة بترحيل البيانات، تقسيم الجدولة، إمكانية توافر عالية وقابلية النسخ، أداء قاعدة البيانات، واستخدام لغات بيرل " Perl" وبايثون " Python" للتكامل مع «بوستجريسكل».

ما النقاط التي يغطيها هذا الكتاب؟

الفصل ١، إدارة قواعد البيانات وخادم «بوستجريسكل»، يساعدك هذا الفصل على إنشاء قواعد البيانات واستيعاب مفهوم المخططات والأدوار والمستخدمين والمجموعات، والجداول في خادم «بوستجريسكل».

الفصل ٢، السيطرة على الأمن، يتيح لك رؤية وفهم الضوابط الأمنية ومستويات الأمن الموجودة في «بوستجريسكل»، فبعد قراءتك لهذا الفصل، يجب أن تكون قادرا على فهم وضبط عناصر التحكم الأمنية الموجودة في خادم ««بوستجريسكل» ويجب أن تكون قادرا أيضا على استخدام اتصالات " طبقة المقابس الآمنة " "SSL" في «بوستجريسكل» من أجل تشفير البيانات.

الفصل ٣، النسخ الاحتياطي والاستعادة ، ويظهر هذا الفصل مختلف سيناريوهات النسخ الاحتياطي والاسترداد التي يمكن تنفيذها في «بوستجريسكل». بعد قراءة هذا الفصل، يجب أن تكون على دراية بطرق النسخ الاحتياطي المنطقي والمادي الملموس واستعادة قواعد البيانات أو كائنات قاعدة البيانات في السيناريو القائم على الاسترداد.

الفصل ٤، يقدم مهام الصيانة الروتينية، معلومات عن الصيانة الدورية للمهام التي تنفذ لتحقيق الأداء الأمثل.

الفصل ٥، مراقبة النظام باستخدام أدوات "يونكس" "Unix"، ويغطي مختلف أوامر "يونكس" "Unix"/ "لينكس" " Linux" التي تفيد في استكشاف أخطاء وحدة المعالجة المركزية "CPU" والذاكرة و مشاكل الإدخال/الإخراج ذات الصلة. بعد قراءة هذا الفصل، يجب أن تكون قادرا على استكشاف الأخطاء وإصلاحها فيما يخص وحدة المعالجة المركزية "CPU"، والذاكرة، ومشاكل تنازع القرص بنجاح باستخدام مختلف أوامر "يونكس" "Unix".

الفصل ٦، مراقبة نشاط قاعدة البيانات والتحقيق في قضايا الأداء، يعلمك الجوانب المختلفة المتعلقة بتحسين أداء «بوستجريسكل». فبعد قراءة هذا الفصل، يجب أن تكون قادرا على حل تناقضات القفل، والعثور على عبارات تشغيل SQL بطيئة ، وجمع الإحصاءات، ودراسة استخدام الفهرس، والتحقيق وحل مشاكل قاعدة بيانات «بوستجريسكل» المختلفة في بيئة الوقت الحقيقي.

الفصل ٧، التوافر العالي وقابلية النسخ، هذا القسم يشرح مفهوم التوافر العالي وقابلية النسخ في «بوستجريسكل». وبعد قراءة هذا الفصل، سوف تكون قادرا على تنفيذ ميزة التوافر العالي وخيارات النسخ باستخدام تقنيات مختلفة بما في ذلك تدفق النسخ، تقنية "سلوني" "Slony" للنسخ المتماثل، النسخ باستخدام "بوكاردو" "Bucardo"، والنسخ باستخدام "لونغديست" "Longdiste". وفي نهاية المطاف، سوف تكون قادرا على تنفيذ جميع خدمات باقة، «بوستجريسكل» النشطة / غير النشطة المتاحة بشكل واسع باستخدام أدوات مفتوحة المصدر مثل "دربد" "DRBD"، "بايسمايكر" "Pacemaker"، و "كوروسينك" " Corosync".

الفصل ٨ ، تجميع الاتصالات، ويتناول أساليب تجميع الاتصالات مثل "بي جي بول" "pgpool" و "بي جي باونس" " Pgbounce". التي تساعد على تقليل الفائض في قاعدة البيانات عندما يكون هناك عدد كبير من الاتصالات المتزامنة. وبعد قراءة هذا الفصل، يجب أن تكون قادرا على إعداد أساليب "بي جي بول" "pgpool" و"بي جي باونس" " Pgbounce".

الفصل ٩، تقسيم الجدول ، ويشرح أساليب التقسيم المختلفة وتنفيذ الفصل المنطقي لبيانات الجدول إلى أقسام. كما يوفر هذا القسم أيضا حصولك على دراية بتطبيق التقسيم الأفقي باستخدام "بي أل/ بروكسي" "PL/Proxy".

الفصل ١٠، النفاذ إلى «بوستجريسكل» من خلال "بيرل" "Perl"، هذا الفصل يجعلك على دراية بإنشاء قاعدة البيانات والاتصالات، والنفاذ إلى البيانات، وأداء عمليات DML على قاعدة بيانات «بوستجريسكل» باستخدام برمجة "بيرل" " Perl".

الفصل ١١، النفاذ إلى «بوستجريسكل» من خلال "بايثون" "Python"، ويظهر لك هذا الفصل كيفية إنشاء الاتصالات الخاصة بقاعدة بيانات والنفاذ إلى البيانات، وتنفيذ عمليات DML على قاعدة بيانات «بوستجريسكل» باستخدام برمجة "بايثون" "Python".،

الفصل ١٢، نقل البيانات من قواعد البيانات الأخرى وتحديث باقة «بوستجريسكل»، ويغطي هذا الفصل التلقائيات المختلفة المتاحة لبدء التحديثات الرئيسية والثانوية لنسخة «بوستجريسكل». كما يزودك هذا الفصل أيضا بطريقة استخدام أداة "أوراكل غولدنغايت" "Oracle GoldenGate" لنسخ البيانات من قواعد البيانات الأخرى إلى «بوستجريسكل».

ما الذي ستحتاجه لهذا الكتاب

ستحتاج إلى البرامج التالية:

- "في أم وير ووورك ستايشن" "VMware Workstation" الإصدار ٧ أو أعلى / "فيرتوال بوكس" "VirtualBox"
- «بوستجريسكل» ٩.٣ مثبت على الجهاز.
- Win32 أوبن أس أس أل الإصدار Win32 OpenSSL v1.0.1" "v1.0.1"
- "بي جي أدمين الإصدار ١.١٨.١" " pgAdmin v1.18.1"
- "بوستغري أس كيو أل" "«بوستجريسكل»" الإصدار ٩.٣
- "أوراكل سولاريس" "Oracle Solaris" الإصدار ١٠
- " سانت أوس لينكس" "CentOS Linux" الإصدار ٦ أو أحدث

من المعني بهذا الكتاب؟

هذا الكتاب هو لمسؤولي النظام، ومسؤولي قاعدة البيانات، والمهندسين المعماريين، والمطورين،وأي شخص لديه اهتمام بالتخطيط والإدارة وتصميم حلول قواعد البيانات باستخدام «بوستجريسكل». هذا الكتاب مثالي لك إذا كان لديك بعض الخبرة السابقة مع أي قاعدة بيانات ذات علاقة أو مع لغة SQL للبرمجة.

الأقسام

يحتوي هذا الكتاب على الأقسام التالية:

الاستعداد للعمل

هذا القسم يخبرنا بما يمكن توقعه في الوصفة، ويصف كيفية إعداد أي برنامج أوأي من الإعدادات الأولية اللازمة لهذه الوصفة.

كيف ينجز ذلك...

يحتوي هذا القسم على الخطوات المطلوبة لمتابعة الوصفة.

كيف تعمل...

يتكون هذا القسم عادة من شرح مفصل لما ذكر في القسم السابق.

هناك المزيد...

يتكون هذا القسم من معلومات إضافية عن الوصفة من أجل جعل القارئ أكثر دراية بها.

انظر أيضا

يوفر هذا القسم روابط مفيدة لمعلومات إضافية حول الوصفة.

الأعراف والأنماط

في هذا الكتاب، سوف تجد عدة أنماط مختلفة للنصوص للتمييز بين أنواع مختلفة من معلومات. وفيما يلي بعض الأمثلة على هذه الأنماط، وتفسيرات لمعانيها.

كلمات الشفرة البرمجية، أسماء جداول قاعدة البيانات، أسماء المجلدات، أسماء الملفات، ملحقات الملفات،

وأسماء المواقع، وعناوين روابط URL ، وإدخال المستخدم، وأمثلة ذلك كما يلي: " الأسلوب الأول يعتمد على استخدام عبارة CREATE DATABASE"

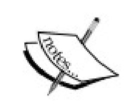
توضع كتلة من التعليمات البرمجية على النحو التالي:

```
SELECT name, setting, unit ,(source = 'default') as is_default
FROM pg_settings WHERE context = 'sighup'
AND (name like '%delay' or name like '%timeout')
AND setting != '0';
```

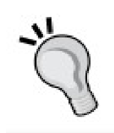
وتكتب سطر أوامر (إدخال أو الإخراج) كما يلي:

```
pg_ctl -D /var/lib/pgsql/data reload
```

مصطلحات جديدة و **كلمات مهمة** تظهر بالخط العريض. كذلك الكلمات التي تراها على الشاشة، في القوائم أو مربعات الحوار على سبيل المثال؛ مثل هذا النص: "في مربع حوار **معالج القاعدة الجديدة الواردة Wizard New Inbound Rule**، انقر على **المراسم والمنافذ Protocol and Ports** ، ثم أنقر على أزرار الانتقاء ،كما هو مبين في الصورة التالية، وأخيرا أنقر على زر **التالي Next**.



تظهر التحذيرات أو الملاحظات المهمة في مربع مثل هذا.



تظهر النصائح والحيل على هذا الشكل.

تعليقات القارئ

تعليقات القراء دائما موضع ترحيب لدينا. فاسمح لنا أن نعرف ما هو رأيك في هذا الكتاب, ما الذي أعجبكم وما لم يعجبكم فيه. فملاحظات القارئ مهمة بالنسبة لنا لتطوير العناوين التي حصلت حقا على أقصى استفادة منها.

ولإرسال التعليقات إلينا، ببساطة قم بإرسال رسالة بريد إلكتروني إلى feedback@packtpub.com، واذكر عنوان الكتاب في موضوع رسالتك.

إذا كان هناك موضوع معين تملك خبرة فيه وكنت مهتما بالكتابة أو المساهمة في الكتاب، راجع دليل المؤلف لدينا على www.packtpub.com/authors.

دعم العملاء

الآن بعد أن أصبحت مالكا فخورا لكتاب من "باكت" "Packt"، لدينا بعض الأمور لمساعدتك في تحقيق أقصى استفادة من الكتاب.

تنزيل رمز المثال

يمكنك تنزيل ملفات شفرة المثال لجميع كتب باكت التي اشتريتها من حسابك على http://www.packtpub.com. إذا قمت بشراء هذا الكتاب من موقع آخر، يمكنك زيارة http://www.packtpub.com/support وسجل للحصول على الملفات بالبريد الإلكتروني مباشرة.

أخطاء مطبعية

على الرغم من أننا قد اتخذنا كل الحرص لضمان دقة المحتوى لدينا، ولكن الأخطاء تحدث.

إذا وجدت خطأ في أحد كتبنا - ربما خطأ في النص أو التعليمات البرمجية – سنكون ممتنين لك إذا بينت هذا الخطأ لنا. فمن خلال ذلك، يمكنك حفظ القراء الآخرين من الإحباط وتساعدنا في تحسين الإصدارات اللاحقة من هذا الكتاب. إذا وجدت أي خطأ، يرجى الإبلاغ عنه من خلال الانتقال إلى http://www.packtpub.com/submit-errata، وتحديد كتابك، النقر على رابط "نموذج تقديم الأخطاء المطبعية" "Errata Submission Form"، وإدخال تفاصيل الخطأ المطبعي. وعندما يتحقق من الخطأ المطبعي، سيقبل التقديم الذي قمت به وسوف يحمل الخطأ إلى موقعنا على الانترنت أو يضاف إلى أي قائمة من الأخطاء الموجودة في قسم الأخطاء المطبعية لنفس العنوان.

لعرض البيانات المقدمة سابقا، انتقل إلى https://www.packtpub.com/books/content/support/ ثم أدخل اسم الكتاب في حقل البحث. المعلومات المطلوبة ستظهر في قسم الأخطاء المطبعية.

القرصنة

إن قرصنة المواد محفوظة الحقوق على الإنترنت هي مشكلة مستمرة في جميع وسائل الإعلام. لكن في "باكت"، نحن نأخذ حماية حقوق التأليف والنشر والتراخيص لدينا على محمل الجد. إذا صادفت أي نسخ غير قانونية من أعمالنا، بأي شكل من الأشكال، على شبكة الإنترنت، يرجى تزويدنا بعنوان أو اسم الموقع فورا حتى نتمكن من معالجة القضية.

يرجى الاتصال بنا على copyright@packtpub.com مع تزويدنا برابط المواد المقرصنة المشتبهة. ونحن نقدر مساعدتك في حماية مؤلفينا، وفي تمكيننا من إنتاج محتوى ثمين.

الأسئلة

يمكنك الاتصال بنا على questions@packtpub.com إذا كنت تواجه مشكلة في أي جانب من الكتاب، ونحن سوف نبذل قصارى جهودنا لمعالجة أي إشكال.

الترجمة العربية

أنتجت الترجمة العربية بواسطة جهود الكثير من المتطوعين، ونحن نتمنى أننا قدمنا ترجمة مقبولة على نطاق واسع عند المختصين، وقليلة الأخطاء سواء أكانت في الترجمة أم إملائية.

إذا صادفت أي أخطاء، الرجاء أخبرنا عنها، للتواصل الرجاء التواصل معنا عبر:

http://itwadi.com/contact

١ - إدارة قواعد البيانات و «بوستجريسكل» الخادم

في هذا الفصل، سنغطي الوصفات التالية:

- إنشاء قواعد بيانات
- إنشاء المخططات
- إنشاء المستخدمين
- إنشاء مجموعات
- تدمير قواعد البيانات
- إنشاء وإسقاط الجداول
- نقل الكائنات بين مساحات الجداول
- إعداد مجموعة قاعدة بيانات
- بدء تشغيل الخادم
- إيقاف الخادم
- عرض حالة الخادم
- إعادة تحميل ملفات إعداد الخادم
- إنهاء الاتصالات

مقدمة

«بوستجريسكل» PostgreSQL هو نظام إدارة غرضي التوجيه لقاعدة البيانات الارتباطية مفتوح المصدر، وتم تطويره أصلا في قسم علوم الكمبيوتر بيريكلي لجامعة كاليفورنيا.

«بوستجريسكل» هو خادم قاعدة بيانات متقدمة متاحة على مجموعة واسعة من المنصات، بدءا من أنظمة التشغيل المستندة إلى يونيكس "Unix-based operating systems" مثل "أوراكل سولاريس" "Oracle Solaris"، و "آي بي إم آيكس" "IBM AIX"، و "آيش بي-يو إكس" "HP-UX"و أنظمة " Windows"؛ و"Mac OS X" و "ريد هات لينكس " "Red Hat Linux" وغيرها من منصات "لينكس" "Linux".

نبدأ أولا بتوضيح كيفية إنشاء قواعد البيانات في «بوستجريسكل». و خلال هذا الفصل، سنغطي المخططات والمستخدمين والمجموعات وأسطح الجداول، ونوضح كيفية إنشاء هذه الكيانات. وسوف نعرض أيضا كيفية بدء خدمات خادم «بوستجريسكل» وإيقافها.

إنشاء قواعد البيانات

قاعدة البيانات هي مجموعة منهجية ومنظمة من البيانات التي يمكن النفاذ إليها بسهولة وإدارتها وتحديثها. فهي تعتبر طريقة فعالة لاسترجاع المعلومات المخزنة. و «بوستجريسكل» هي قاعدة بيانات قوية و مفتوحة. وهي قابلة للنقل لأنها مكتوبة على "أنسي سي" "ANSI C " ونتيجة لذلك، فهي متاحة لمنصات مختلفة وموثوق بها. كما أنها متوافقة مع الـ " **ACID**" كاختصار لـ **Atomicity**, **Consistency**, **Isolation**, **Durability**)) بمعنى **(قابلية التجزئة و التناسق و قابلية العزل و المتانة)** كما أنها تدعم المعاملات وقابلة للتطوير لأنها تدعم التحكم في **تعدد النسخ المتزامن (MVCC)** وعملية تقسيم الجدولة فيها آمنة لأنها توظف التحكم في النفاذ استنادا إلى المضيف وتدعم " طبقة المقابس الآمنة " "SSL" ، و تتيح توافر عالي وقابلية نسخ عن طريق تحقيق بعض الامتيازات كبث النسخ المتعدد ودعمها لنقطة استعادة الوقت.

الاستعداد للعمل

قبل البدء في إنشاء قواعد البيانات، سوف تحتاج إلى تثبيت «بوستجريسكل» على جهاز الكمبيوتر الخاص بك.

بالنسبة لبيئات "ريد هات" "Red Hat " أو "سينت أوس لينوكس" "CentOS Linux" ، يمكنك تنزيل rpm الصحيحة لنسخة «بوستجريسكل ٩.٣» من خلال yum.postgresql.org

هذا هو الرابط الذي يمكنك استخدامه لتثبيت «بوستجريسكل» على "سينت أوس " " CentOS ":

```
http://www.postgresonline.com/journal/archives/329-An-almost-idiotsguide-to-install-PostgreSQL-9.3,-PostGIS-2.1-and-pgRouting-with-Yum.html
```

أما الروابط التالية يمكنك استخدامها لتثبيت «بوستجريسكل» على منصة "أبونتو " "Ubuntu":

```
http://technobytz.com/install-postgresql-9-3-ubuntu.html
http://www.cloudservers.com/installing-and-configuringpostgresql-9-3-on-hosted-linux-cloud-vps-server/
```

بدلا من ذلك، يمكنك تحميل المثبت «بوستجريسكل» الرسومي المتاحة من موقع "إنتيربرايس دي بي " "EnterpriseDB"، على هذا الرابط

```
http://www.enterprisedb.com/products-servicestraining/pgdownload
```

للحصول على تفاصيل حول كيفية تثبيت «بوستجريسكل» باستخدام المثبت «بوستجريسكل» الرسومي من موقع "إنتيربرايس دي بي" يمكنك الرجوع إلى الرابط التالي للحصول على مزيد من التعليمات:

```
http://www.enterprisedb.com/docs/en/9.3/pginstguide/Table%20of%20Contents.htm
```

وبمجرد الانتهاء من تحميل وتثبيت «بوستجريسكل»، سوف تحتاج إلى تحديد موقع البيانات، وهو موقع تخزين كافة ملفات البيانات لقاعدة البيانات. ثم سوف تحتاج إلى إعداد دليل البيانات. ويتم تضمين إعداد دليل البيانات تحت وصفة بعنوان " Initializing a database cluster" أي "إعداد كتلة قاعدة بيانات". بعد ذلك، ستكون على استعداد لإنشاء قاعدة البيانات.

للاتصال بقاعدة بيانات باستخدام خدمة psql، يمكنك استخدام الأمر التالي:

```
psql -h localhost -d postgres —p 5432
```

هنا، نحن في الأساس على اتصال مع قاعدة بيانات "postgres" ، التي تقيم على استضافة محلية، وهذا هو نفس الخادم الذي قام بتثبيت «بوستجريسكل»، والاتصال يجري على المنفذ ٥٤٣٢.

في التعليمات البرمجية التالية، نحن نقوم بخلق مستخدم "hr" في الأساس، يتم إنشاء هذا المستخدم لأنه في القسم التالي، يتم استخدامه كمالك لقاعدة بيانات "hrdb" :

```
CREATE USER hr with PASSWORD 'hr';
```

ستتم تغطية المزيد من التفاصيل المتعلقة بإنشاء المستخدمين في وصفة إنشاء المستخدمين.

كيف ينجز ذلك...

- يوفر «بوستجريسكل» طريقتين لإنشاء قاعدة بيانات جديدة:

تعتمد الطريقة الأولى على استخدام عبارة س.ك.ل "CREATE DATABASE" :

```
CREATE DATABASE hrdb WITH ENCODING='UTF8' OWNER=hr CONNECTION LIMIT=25;
```

- الأسلوب الثاني يتطلب استخدام سطر الأوامر "createdb" القابل للتنفيذ:

```
createdb —h localhost —p 5432 —U postgres testdb1
```

كيف تعمل...

قاعدة البيانات عبارة عن مجموعة من الكائنات كالجداول والوظائف وما إلى ذلك.

من أجل إنشاء قاعدة بيانات، يجب أن يكون المستخدم إما مستخدم جذر أو يجب أن يحصل على امتياز " CREATEDB" خاص.

يتم إنشاء سطر الأوامر القابلة للتنفيذ "createdb" المتصلة بقاعدة بيانات "postgres"، ثم يتم إصدار الأمر " CREATE DATABASE".

يمكنك عرض قائمة قواعد البيانات الموجودة عن طريق الاستعلام عن الجدول " pg_databasecatalog"، كما هو مبين في الصورة التالية:

```
postgres=# SELECT datname from pg_database  WHERE datistemplate = false;
 datname
----------
 postgres
 testdb1
 hrdb
(3 rows)
```

أو، يمكنك استخدام مبدل l/ من psql لعرض قائمة قواعد البيانات الموجودة.

إنشاء المخططات schemas

المخططات schemas تعتبر من أهم الكائنات داخل قاعدة البيانات. و**المخطط** عبارة عن مجموعة من الجداول تحمل تسمية. كما قد يحتوي المخطط أيضا على مشاهدات وفهارس وسلاسل وأنواع البيانات والعمال والوظائف. المخططات

تساعد في تنظيم كائنات قاعدة البيانات إلى مجموعات منطقية، وهذا بدوره يساعد على جعل هذه الكائنات أكثر قابلية للإدارة.

كيف ينجز ذلك...

يمكنك استخدام تصريح "CREATE SCHEMA" لإنشاء مخطط جديد في «بوستجريسكل»:

```
CREATE SCHEMA employee;
```

أو، من الممكن أيضا إنشاء مخطط لمستخدم معين:

```
CREATE SCHEMA university AUTHORIZATION bob;
```

في هذا المثال، المخطط يسمى "university" و قد تم إنشاء هذا المخطط وجعل ملكيته لـ"bob".

كيف تعمل...

المخطط هو كيان منطقي يساعد على تنظيم الكائنات والبيانات في قاعدة البيانات. وفرضًا، إذا لم تقم بإنشاء أية مخططات، لن يتم إنشاء أي كائنات جديدة في المخطط العام.

ومن أجل إنشاء مخطط، يجب أن يكون المستخدم إما مستخدم جذر أو يجب أن يحصل على امتياز "CREATE" لقاعدة البيانات الحالية.

بمجرد إنشاء مخطط، يمكن استخدامه لإنشاء كائنات جديدة مثل الجداول و المشاهدات ضمن هذا المخطط.

هناك المزيد...

يمكنك استخدام مبدل \dn من psql لعرض كافة المخططات في قاعدة بيانات كما هو مبين في الصورة:

```
postgres=# \dn
  List of schemas
  Name   |  Owner
--------+----------
 hr      | postgres
 hrd     | hr
 public  | postgres
(3 rows)
```

لتحديد المخطط الذي تعمل فيه حاليا، يمكنك استخدام الأمر التالي:

```
SELECT current_schema();
```

أثناء البحث عن الكائنات الموجودة في قاعدة البيانات، يمكنك تحديد تفضيلات مخطط البحث للموقع الذي يجب أن تبدأ فيه عمليات البحث هذه. يمكنك استخدام خواص مسار البحث search_path لهذا، كما يلي:

```
ALTER DATABASE hrd SET search_path TO hr,hrms, public, pg_catalog;
```

إنشاء المستخدمين

المستخدم هو دور ولوج الذي يسمح بالولوج إلى خادم «بوستجريسكل». قسم أدوار الولوج هو الموقع الذي تحدد فيه حسابات المستخدمين الفرديين لنظام «بوستجريسكل». يجب أن يكون لكل مستخدم قاعدة بيانات حساب فردي للولوج إلى نظام «بوستجريسكل». كل مستخدم لديه معرف داخلي في «بوستجريسكل»، يعرف باسم sysid. يتم استخدام معرف النظام الخاص بالمستخدم لربط الكائنات في قاعدة بيانات مع مالكها. قد يكون للمستخدمين أيضا حقوق عالمية مخصصة لهم عند إنشائها. تحدد هذه الحقوق ما إذا كان يسمح للمستخدم بإنشاء قواعد بيانات أو إسقاطها وما إذا كان المستخدم الحالي مستخدما متميزا أم لا.

كيف ينجز ذلك...

يوفر «بوستجريسكل» طريقتين يتم من خلالها إنشاء مستخدمي قاعدة البيانات:

١- الطريقة الأولى تتطلب استخدام عبارة س.ك.ل CREATE USER

لإنشاء مستخدم جديد في قاعدة البيانات. يمكنك إنشاء مستخدم جديد باستخدام عبارة س.ك.ل "CREATE USER" كما يلي:

```
CREATE user agovil WITH PASSWORD 'Kh@rt0um';
```

هنا، أنشأنا "agovil" وقدمنا كلمة مرور للمستخدم لتسجيل الدخول بها.

٢- الطريقة الثانية تتطلب تنفيذ سكريبت "createuser" من سطر الأوامر.

قد نستخدم أيضا سكريبت createuser لإنشاء مستخدم يسمى nchabbra على نفس المضيف (المنفذ ٥٤٣٢)، و يحدد الخيار S- أن المستخدم الذي تم إنشاؤه لن يتمتع بامتيازات المستخدم الجذر:

```
$ createuser -h localhost -p 5432 -S nchabbra
```

كيف تعمل...

تتطلب عبارة س.ك.ل "CREATE USER" معامل إلزامي واحد هو اسم المستخدم الجديد. ومع ذلك، هناك معاملات أخرى اختيارية هي كلمات مرور للمستخدم أو المجموعة ومعرف النظام ومجموعة من الامتيازات التي قد يتم تخصيصها بشكل صريح.

يمكن استدعاء سكريبت "createuser" بدون المعطيات . وفي هذه الحالة، سوف يجبرنا على تقديم اسم المستخدم ومجموعة الحقوق وستتم محاولة إجراء اتصال محلي إلى «بوستجريسكل» . ويمكن أيضا استدعاؤه مع الخيارات واسم المستخدم ليتم إنشاؤه على سطر الأوامر، وسوف تحتاج إلى إعطاء المستخدم إمكانية النفاذ إلى قاعدة البيانات بشكل صريح إذا كان المستخدم ليس مالكا لقاعدة البيانات.

هناك المزيد...

يمكننا استخدام مبدل \du من psql لعرض قائمة المستخدمين الحاليين، بما في ذلك الأدوار في خادم «بوستجريسكل»، كما هو مبين في هذه الصورة:

```
postgres-# \du
                              List of roles
   Role name   |                 Attributes                   |   Member of
---------------+----------------------------------------------+--------------
--
 agovil        |                                              | {dba_community
}
 dba_community | Cannot login                                 | {}
 hr            | Cannot login                                 | {}
 manager       | Superuser                                    | {}
 nchabbra      |                                              | {dba_community
}
 nchabra       |                                              | {}
 postgres      | Superuser, Create role, Create DB, Replication | {}
 salesuser     |                                              | {}
```

أو يمكنك الحصول على قائمة المستخدمين عن طريق الاستعلام عن جدول pg_user باستخدام س.ك.ل ، كما هو موضح في لقطة الشاشة التالية:

```
postgres=# SELECT u.usename AS "User name",u.usesysid AS "User ID",CASE WHEN u.usesuper AND u.usecreatedb THEN CAST('superuser, create database' AS pg_catalog.text) WHEN u.usesuper THEN CAST('superuser' AS pg_catalog.text) WHEN u.usecreatedb THEN CAST('create database' AS pg_catalog.text) ELSE CAST('' AS pg_catalog.text) END AS "Attributes" FROM pg_catalog.pg_user u ORDER BY 1;
 User name | User ID |         Attributes
-----------+---------+---------------------------
 agovil    |   16399 |
 manager   |   16400 | superuser
 nchabbra  |   16403 |
 nchabra   |   16402 |
 postgres  |      10 | superuser, create database
 salesuser |   16398 |
(6 rows)
```

إنشاء مجموعات

المجموعة في خادم «بوستجريسكل» تشبه المجموعات الموجودة في يونكس و لينكس. والمجموعة في «بوستجريسكل» تساعد في تبسيط التنازل عن الحقوق. ببساطة تحتاج إلى اسم ويمكن إنشاؤها فارغة. وبمجرد إنشاء المجموعة، يتم إضافة المستخدمين الذين يقصد بهم مشاركة حقوق النفاذ المشتركة إلى المجموعة معا، وبالتالي يرتبطون بعضوية هذه المجموعة. ثم تمنح صلاحيات كائنات قاعدة البيانات إلى المجموعة بدلا من كل عضو على حدة.

كيف ينجز ذلك...

يمكن إنشاء المجموعات في خادم «بوستجريسكل» باستخدام عبارة س.ك.ل "CREATE GROUP". وسيقوم الأمر التالي بإنشاء مجموعة. رغم عدم وجود مستخدمين حاليا في هذه المجموعة:

```
hrdb=# CREATE GROUP dept;
```

من أجل تضمين الأعضاء / المستخدمين إلى المجموعة، يمكننا استخدام عبارة "ALTER GROUP" على النحو التالي:

```
hrdb=# ALTER GROUP dept ADD USER agovil,nchabbra;
```

ومن الممكن أيضا إنشاء مجموعة وتعيين المستخدمين عند إنشائها، كما هو مبين في عبارة "CREATE GROUP" التالية:

```
hrdb=# CREATE GROUP admins WITH user agovil,nchabbra;
```

كيف تعمل...

المجموعة عبارة عن كائن قاعدة بيانات على مستوى النظام يمكن تعيين لها امتيازات ولها مستخدمين يتم إضافتهم كأعضاء.

المجموعة هي دور لا يمكن استخدامه لتسجيل الدخول إلى أي قاعدة بيانات كانت.

ومن الممكن أيضا منح العضوية في مجموعة إلى مجموعة أخرى، مما يسمح باستخدام دور العضو للامتيازات المعينة للمجموعة التي هي عضو فيها.

مجموعات قاعدة البيانات هي عالمية عبر قاعدة بيانات ذات العناقيد.

هناك المزيد...

لسرد كافة المجموعات المتوفرة في حالة خادم «بوستجريسكل»، تحتاج إلى الاستعلام عن جدول pg_group ، كما هو مبين في الصورة التالية:

```
hrdb=# SELECT * FROM pg_group;
    groname      | grosysid |     grolist
-----------------+----------+---------------
 hr              |    16394 | {}
 dba_community   |    16404 | {16399,16403}
 sam             |    16418 | {16399,16417}
 sales           |    16419 | {}
 data            |    16420 | {}
 dept            |    16421 | {16399,16403}
(6 rows)
```

تدمير قواعد البيانات

كل مزود رئيسي لـ "RDBMS" -وهي اختصار لـ " نظام إدارة قواعد البيانات الارتباطية"- يوفر القدرة على حذف قواعد البيانات تماما كما أنها تسمح لك بخلق قواعد بيانات جديدة. ومع ذلك، ينبغي للمرء توخي الحذر عند التعامل مع هذه الحالات أي حذف قواعد بيانات. ففي حالة حذف قاعدة بيانات، تفقد جميع المعلومات الموجودة فيها إلى الأبد.

تحذف قاعدة البيانات فقط في حالة وجود غرض تجاري مقبول. وفي الظروف العادية، تحذف قاعدة البيانات فقط عندما يتم إيقاف تشغيلها ولم تعد هناك حاجة للعمليات التجارية.

كيف ينجز ذلك...

هناك طريقتان لحذف قاعدة بيانات في حالة خادم «بوستجريسكل»:

- يمكنك استخدام عبارة "DROP DATABASE" لإسقاط قاعدة بيانات من «بوستجريسكل»، على النحو التالي:

```
hrdb=# DROP DATABASE hrdb;
```

- يمكنك استخدام أمر " dropdb" وهو غلاف حول DATABASE DROP

```
$ dropdb hrdb;
```

كيف تعمل...

تقوم عبارة "DROP DATABASE" بحذف مدخلات الفهرس ودليل البيانات نهائيا.

يمكن فقط لمالك قاعدة البيانات إصدار عبارة "DROP DATABASE".

أيضا، ليس من الممكن حذف قاعدة بيانات و أنت متصل بها. لحذف قاعدة بيانات، يجب على مالكها إجراء اتصال بقاعدة بيانات أخرى .

هناك المزيد...

إحدى الحالات التي تتطلب الانتباه هي عندما يحاول المستخدم حذف قاعدة بيانات ذات اتصالات نشطة. سوف يحصل المستخدم على خطأ عند محاولة حذف قاعدة البيانات هذه.

لحذف قاعدة بيانات لها اتصالات نشطة معها، سيكون عليك اتباع الخطوات التالية:

١- تحديد جميع الجلسات النشطة على قاعدة البيانات. لتحديد كل الجلسات النشطة على قاعدة البيانات، تحتاج إلى الاستعلام عن الجدول التالي pg_stat_activity كما يلي:

```
SELECT * from pg_stat_activity where datname='testdb1';
```

٢- قم بإنهاء كافة الاتصالات النشطة إلى قاعدة البيانات. لإنهاء جميع الاتصالات النشطة، سوف تحتاج إلى استخدام وظيفة pg_terminate_backend كما يلي:

```
SELECT pg_terminate_backend(pid) FROM pg_stat_activity WHERE
datname = 'testdb1';
```

٣- بمجرد إنهاء كافة الاتصالات، يمكنك المتابعة لحذف قاعدة البيانات باستخدام DROP DATABASE.

إنشاء وحذف مساحة جدول

تقوم «بوستجريسكل» بتخزين ملفات البيانات التي تتكون من كائنات قاعدة البيانات مثل الجداول والمؤشرات على القرص. يتم تعريف " مساحة الجدول " "tablespace" باسم موقع هذه الكائنات على القرص. **مساحة الجدول** تستخدم لتعيين اسم منطقي لموقع فعلي على القرص.

الاستعداد للعمل

مساحة الجدول هي موقع تخزين ملفات البيانات التي تحتوي على كائنات قاعدة بيانات في القرص الصلب حيث تقوم «بوستجريسكل» بتخزين على سبيل المثال الفهارس والجداول، وهلم جرا.

قبل إنشاء مساحة الجداول، يجب إنشاء موقع الدليل فعليا ويجب أن يكون الدليل فارغا:

```
mkdir –p /var/lib/pgsql/data/dbs
```

كيف ينجز ذلك...

لإنشاء مساحة الجداول في بوستجريسكل، تحتاج إلى استخدام "CREATE TABLESPACE".

يؤدي الأمر التالي إلى إنشاء مساحة الجداول data_tbs، الذي يملكه المستخدم "agovil":

```
CREATE TABLESPACE data_tbs OWNER agovil LOCATION
'/var/lib/pgsql/data/dbs';
```

وبالمثل، يمكن حذف مساحة الجداول في بوستجريسكل باستخدام "DROP TABLESPACE"، على النحو التالي:

```
DROP TABLESPACE data_tbs;
```

كيف تعمل...

مساحة الجداول تسمح لك بالسيطرة على تخطيط القرص الصلب لبوستجريسكل. مالك مساحة الجداول سيكون بشكل افتراضي المستخدم الذي قام بتنفيذ "CREATE TABLESPACE". يمنحك هذا البيان أيضا خيار تعيين ملكية مساحة الجدول إلى مستخدم جديد. هذا الخيار هو جزء من بند الملكية "owner" في عبارة "CREATE TABLESPACE".

اسم مساحة الجدول يجب أن لا تبدأ بـ pg_prefix لأن هذا محجوز لمساحات جداول النظام.

قبل حذف مساحة الجداول، تأكد من أنها فارغة، مما يعني أنه يجب عدم وجود كائنات قاعدة بيانات داخلها. إذا حاول المستخدم حذف مساحة الجدول غير فارغ فسيخفق الأمر.

هناك خياران سيساعدان في حذف مساحة الجدول عندما لا يكون فارغا:

- يمكنك حذف قاعدة البيانات
- يمكنك تغيير قاعدة البيانات لنقلها إلى مساحة مختلفة.

بعد الانتهاء من أي من الإجراءات السابقة، فإنه يصبح من الممكن حذف مساحة الجداول المعنية.

هناك المزيد...

افتراضيا، يوجد جدولان في بوستجريسكل:

- pg_default: يستخدم لتخزين بيانات المستخدم.
- pg_global: يستخدم هذا لتخزين البيانات الإجمالية.

يمكنك الاستعلام عن الجدول pg_tablespacecatalog للحصول على قائمة مساحات الجداول الموجودة في بوستجريسكل، كما هو موضح في لقطة الشاشة التالية:

```
postgres=# select * from pg_tablespace;
  spcname   | spcowner | spcacl | spcoptions
------------+----------+--------+------------
 pg_default |       10 |        |
 pg_global  |       10 |        |
 demo       |       10 |        |
(3 rows)
```

نقل الكائنات بين مساحات الجداول

يمكن أن تحتوي مساحة الجداول على كائنات دائمة ومؤقتة. سوف تحتاج إلى تعريف وإنشاء مساحة جدول ثانوية لتكون الوجهة المستهدفة للكائنات التي قد يتم نقلها من مساحة الجداول الأساسية. نقل الكائنات بين مساحات الجداول هو آلية نسخ البيانات المجمعة حيث تنسخ بالتتابع، كتلة بكتلة. يؤدي نقل جدول إلى مساحة جدول آخر إلى تأمينه خلال مدة الانتقال.

الاستعداد للعمل

هنا، سنقوم أولا بإنشاء مساحة جداول جديدة، "hrms"، باستخدام الأمر التالي:

```
mkdir –p /var/lib/pgsql/data/hrms
```

ثم نقوم بتعيين مساحة الجداول الافتراضية لقاعدة البيانات testdb1 إلى "hrms" باستخدام العبارة التالية:

```
CREATE TABLESPACE HRMS OWNER agovil LOCATION
'/var/lib/pgsql/data/hrms';
```

سنقوم أيضا بإنشاء جدول، وإدراج بعض السجلات في ذلك، وإنشاء فهرس المقابلة لذلك. كل هذا لأنه سيتم استخدام الجدول ومؤشره في قسم "كيف ينجز ذلك..." من هذه الوصفة:

```
CREATE TABLE EMPLOYEES(id integer PRIMARY KEY , name varchar(40));
INSERT INTO EMPLOYEES VALUES (1, 'Mike Johansson');
INSERT INTO EMPLOYEES VALUES(2, 'Rajat Arora');
CREATE INDEX emp_idx on employees(name);
```

كيف ينجز ذلك...

نقل قاعدة بيانات كاملة إلى مساحة جدول مختلفة تتضمن ثلاث خطوات:

١- سوف تقوم بتغيير مساحة الجداول لقاعدة البيانات المحددة بحيث يتم إنشاء كائنات جديدة لقاعدة البيانات المقترنة في مساحة الجداول الجديدة:

```
ALTER DATABASE testdb1 SET default_tablespace='hrms';
```

٢- ثم يستوجب عليك نقل كافة الجداول الموجودة في قاعدة البيانات المقابلة إلى مساحة الجداول الجديدة:

```
ALTER TABLE employee SET TABLESPACE hrms;
```

٣- سيكون عليك أيضا نقل كل الفهارس الموجودة إلى مساحة الجداول الجديدة:

```
ALTER INDEX emp_idx SET TABLESPACE hrms;
```

كيف تعمل...

سيكون عليك الاستعلام عن الجدول pg_tables لمعرفة الجداول التي تحتاج إلى نقلها من قاعدة البيانات الحالية إلى مساحة جدول أخرى.

وبالمثل بالنسبة للمؤشرات، سيكون عليك الاستعلام عن الجدول pg_indexes لمعرفة الفهارس التي تحتاج إلى نقلها إلى مساحة جدول أخرى.

تهيئة عنقود قواعد بيانات

في اصطلاح نظام الملفات، **عنقود قاعدة بيانات** "a database cluster" هي مجموعة من قواعد البيانات التي يتم إدارتها بواسطة خادم واحد، وهو الإطار الذي أنشئت عليه قاعدة بيانات بوستجريسكل.

كيف ينجز ذلك...

يتم استخدام الأمر "initdb" لتهيئة أو إنشاء كتلة قاعدة البيانات. ويستخدم مبدل D- لأمر "initdb" لتحديد موقع نظام ملفات عنقود قاعدة البيانات.

لإنشاء عنقود قاعدة البيانات استخدم الأمر "initdb":

```
$ initdb -D /var/lib/pgsql/data
```

طريقة أخرى لتهيئة عنقود قواعد البيانات هي عن طريق استدعاء الأمر "initdb" عبر وسيلة pg_ctl

```
$ pg_ctl -D /var/lib/pgsql/data initdb
```

كيف تعمل...

عنقود قواعد البيانات هي عبارة عن مجموعة من قواعد البيانات التي تدار بواسطة خادم واحد.

عند تشغيل الأمر "initdb"، سيتم إنشاء أدلة المجلدات التي ستضع بها بيانات قاعدة البيانات، و يتم إنشاء جداول الكتالوج المشتركة، ويتم إنشاء قاعدة بيانات template1 و قاعدة بيانات postgres ؛ التي تكون منها قاعدة البيانات الافتراضية postgres. كذلك يقوم الأمر "initdb" بتهيئة اللغة الافتراضية لعنقود قواعد البيانات وترميز مجموعة الأحرف.

يمكنك الرجوع إلى http://www.postgresql.org/docs/9.3/static/creatingcluster.html لمزيد من المعلومات حول تهيئة عنقود قواعد بيانات.

بدء تشغيل الخادم

قبل أن يتمكن أي شخص من النفاذ إلى قاعدة البيانات، يجب بدء تشغيل خادم قاعدة البيانات. ثم ستكون قادرا على بدء تشغيل كافة الحالات من قاعدة البيانات postgres في العنقود باستخدام أوامر مختلفة مع خيارات كما هو مذكور في هذه الوصفة.

الاستعداد للعمل

مصطلح "الخادم" يشير إلى قاعدة البيانات وعمليات الخلفية المرتبطة بها. يشير مصطلح "خدمة" "service" إلى ملف نظام التشغيل الذي يتم من خلاله استدعاء الخادم. في الظروف العادية، خادم بوستجريسكل عادة ما يبدأ تلقائيا عندما يتم تشغيل النظام. ومع ذلك، سيكون هناك حالات حيث قد يكون عليك بدء تشغيل الخادم يدويا لأسباب مختلفة.

كيف ينجز ذلك...

هناك طريقتان يمكن من خلالها بدء تشغيل خادم بوستجريسكل على منصات يونكس أو لينكس:

- تعتمد الطريقة الأولى على تمرير مُعطى البداية إلى وسيلة pg_ctl للحصول على بدء عملية postmaster الخلفية، الذي يعني فعلياً بدء تشغيل خادم بوستجريسكل.
- تعتمد الطريقة التالية على استخدام أوامر الخدمة، بحيث، إذا كانت مدعومة من قبل نظام التشغيل، يمكن أن تستخدم كغلاف للبرنامج النصي المثبت بوستجريسكل.
- الأسلوب الأخير ينطوي على استدعاء السكريبت بوستجريسكل المثبت مباشرة باستخدام مساره الكامل.

في معظم توزيعات يونكس وتوزيعات لينكس المستندة إلى ريد هات، يمكن استخدام وسيلة pg_ctl على النحو التالي:

```
pg_ctl -D /var/lib/pgsql/data start
```

إذا كنت تستخدم أمر الخدمة، يمكن بدء تشغيل الخدمة كما يلي:

```
service postgresql<version> start
```

لبوستجريسكل الإصدار ٩.٣، الأمر خدمة لبدء خادم بوستجريسكل يكون كما يلي:

```
service postgresql-9.3 start
```

يمكنك أيضا بدء تشغيل الخادم عن طريق استدعاء السكريبت بوستجريسكل المثبت يدويا باستخدام مساره الكامل:

```
/etc/rc.d/init.d/postgresql-9.3 start
```

على الأنظمة الخاصة بالويندوز، يمكن بدء خدمة بوستجريسكل باستخدام الأمر التالي:

```
NET START postgresql-9.3
```

كيف تعمل...

وسيطة البداية لوسيلة pg_ctl تقوم أولا ببدء تشغيل عملية postmaster الخلفية لبوستجريسكل باستخدام مسار دليل البيانات.

بعد ذلك سيشتغل نظام قاعدة البيانات بنجاح، وسيبلغ عن آخر مرة تم إيقاف نظام قاعدة البيانات، و مختلف عبارات التنقيح قبل إرجاع مستخدم postgres إلى محث صدفة shell.

هناك المزيد...

في توزيعات أبونتو "Ubuntu" و ديبيان لينكس"Debian Linux"، يمكن استخدام غلاف pg_ctlcluster مع معطى start لبدء خادم postmaster لعنقود معين. العنقود عبارة عن مجموعة من واحد أو أكثر من خوادم قاعدة البيانات بوستجريسكل التي قد تتواجد على مضيف واحد.

إيقاف الخادم

في بعض الأحيان في حالات الطوارئ، قد تضطر إلى إيقاف خدمات خادم بوستجريسكل. هناك بعض الحالات التي قد تحتاج إلى إيقاف خدمات قاعدة البيانات.

على سبيل المثال، أثناء عملية ترحيل نظام التشغيل، قد تحتاج إلى إيقاف تشغيل الخدمات وأخذ نسخة احتياطية من نظام الملفات ثم المتابعة ترحيل نظام التشغيل.

كيف ينجز ذلك...

هناك طريقتان يمكن من خلالها إيقاف خادم بوستجريسكل.

على توزيعات يونكس وتوزيعات لينكس المستندة إلى ريد هات، يمكننا استخدام معطى stop من وسيلة pg_ctl لإيقاف postmaster:

```
pg_ctl -D /var/lib/pgsql/data stop -m fast
```

باستخدام أمر الخدمة، يمكن إيقاف خادم بوستجريسكل مثل هذا:

```
service postgresql stop
```

يمكنك أيضا إيقاف الخادم عن طريق استدعاء برنامج بوستجريسكل النصي المثبت يدويا باستخدام مساره الكامل:

```
/etc/rc.d/init.d/postgresql stop
```

على الأنظمة المستندة إلى ويندوز، يمكنك إيقاف خدمة postmaster بهذه الطريقة:

```
NET STOP postgresql-9.3
```

كيف تعمل...

تفحص أداة pg_ctl عملية postmater الحالية، وإذا تم استدعاء معطى stop لأداة pg_ctl سيتم إيقاف الخادم.

افتراضيا، سوف يقوم خادم بوستجريسكل أولا بانتظار إلغاء العملاء اتصالاتهم قبل إيقاف تشغيل.

وعلى كل حال إذا استخدام الإغلاق السريع، لن يمنح وقت انتظار وستحبط كافة معاملات المستخدم وتقطع جميع الاتصالات.

هناك المزيد...

قد تكون هناك حالات حيث يحتاج المرء إلى إيقاف خادم بوستجريسكل في حالة الطوارئ، ولهذا، يوفر بوستجريسكل وضع إيقاف التشغيل الفوري.

في حالة الإغلاق الفوري، سوف تتلقى العملية إشارة شديدة ولن تكون قادرة على الاستجابة إلى الخادم بعد ذلك.

نتيجة هذا النوع من الإغلاق هو أن بوستجريسكل لن يكون قادر على الانتهاء من عمليات الإدخال والإخراج للقرص الصلب ، وبالتالي يجب أن تقوم الاستعادة من الانهيار في المرة القادمة التي يتم فيها تشغيله.

يمكن استدعاء وضع الإغلاق الفوري كما يلي:

```
pg_ctl -D /var/lib/pgsql/data stop -m immediate
```

وهناك طريقة أخرى لإغلاق الخادم سيكون بإرسال إشارة مباشرة باستخدام الأمر قتل kill . يمكن العثور على رمز المهمة PID للإجراء postgres باستخدام الأمر ps أو في ملف postmaster.pid في دليل data. من أجل بدء الإغلاق السريع، يمكنك إصدار الأمر التالي:

```
$ kill -INT head -1 /usr/local/pgsql/data/postmaster.pid
```

عرض حالة الخادم

في كثير من الأحيان، سيكون هناك حالات حيث يشكو المستخدمين النهائيين أن أداء قاعدة البيانات بطيء وأنها ليست قادرة على تسجيل الدخول إلى قاعدة البيانات. في مثل هذه الحالات، غالبا ما يكون من المفيد أن نلقي نظرة سريعة على حالة إجراء postmaster مشغل بوستجريسكل وتأكيد ما إذا كانت خدمات خادم بوستجريسكل قيد التشغيل.

كيف ينجز ذلك...

هناك طريقتان يمكن من خلالها التحقق من حالة خادم بوستجريسكل.في يونكس وعلى توزيعات لينكس المستندة إلى ريد هات، يمكن استخدام معطى وسيلة pg_ctl للتحقق من حالة تشغيل خلفية "postmaster":

```
pg_ctl -D /var/lib/pgsql/data status
```

في المنصات التي تعتمد أساسا على يونكس و لينكس والتي تدعم أمر الخدمة ، يمكن التحقق من حالة إجراء " postgresql" كما يلي:

```
service postgresql status
```

يمكنك أيضا التحقق من حالة الخادم عن طريق استدعاء يدويا برنامج بوستجريسكل المثبت باستخدام مساره الكامل:

```
/etc/rc.d/init.d/postgresql status
```

كيف تعمل...

وضع حالة أداة pg_ctl تتحقق ما إذا كان إجراء postmaster قيد التشغيل في دليل البيانات المحدد.

إذا كان الخادم قيد التشغيل سيتم عرض معرف الإجراء وخيارات سطر الأوامر التي تم استخدامها لاستدعائه.

إعادة تحميل ملفات ضبط الخادم

التغييرات التي يتم إجراؤها على بعض معاملات ضبط بوستجريسكل تصبح نافذة المفعول عند إعادة تحميل ملفات ضبط الخادم مثل postgresql.conf ، إعادة تحميل ملفات ضبط الخادم يصبح ضروريا في لهذه الحالات.

كيف ينجز ذلك...

بعض معاملات ضبط في بوستجريسكل يمكن تغييرها على السريع. ومع ذلك، التغييرات على تضبيطات أخرى يمكن أن تنعكس فقط بمجرد إعادة تحميل ملفات ضبط الخادم.

على معظم منصات يونكس المستندة إلى لينكس، يكون الأمر بإعادة تحميل ملف ضبط الخادم كما يلي:

```
pg_ctl -D /var/lib/pgsql/data reload
```

من الممكن أيضا إعادة تحميل ملف الضبط أثناء الاتصال بجلسة بوستجريسكل. لكن يمكن أن يتم ذلك من قبل المستخدم الجذري فقط:

```
postgres=# select pg_reload_conf();
```

على ريد هات والأنظمة الأخرى المستندة إلى لينوكس التي تدعم الأمر خدمة,الأمر بوستجريسكل لإعادة تحميل ملف الضبط يكون كما يلي:

```
service postgresql reload
```

كيف تعمل...

للتأكد من أن التغييرات التي تم إجراؤها على المعاملات في ملف ضبط نافذة المفعول؛ يلزم إعادة تحميل ملف الضبط. تتطلب إعادة تحميل ملفات الضبط إرسال إشارة "sighup" إلى إجراء "postmaster"، والتي بدورها ستعيد توجيهها إلى جلسات الخلفية الأخرى المتصلة.

هناك بعض معلمات الضبط التي يمكن أن تنعكس فيها القيم التي تم تغييرها فقط من خلال إعادة تحميل الخادم. تحتوي معاملات الضبط هذه على قيمة تعرف باسم "sighup" في سياق السمة في جدول الكتالوج pg_settings:

```
SELECT name, setting, unit ,(source = 'default') as is_default FROM
pg_settings WHERE context = 'sighup'
AND (name like '%delay' or name like '%timeout')
AND setting != '0';
```

الإخراج للاستعلام السابق يكون كما هو موضح أدناه:

```
            name              | setting | unit | is_default
------------------------------+---------+------+------------
 authentication_timeout       | 60      | s    | t
 autovacuum_vacuum_cost_delay | 20      | ms   | t
 bgwriter_delay               | 200     | ms   | t
 checkpoint_timeout           | 300     | s    | t
 max_standby_archive_delay    | 30000   | ms   | t
 max_standby_streaming_delay  | 30000   | ms   | t
 wal_receiver_timeout         | 60000   | ms   | t
 wal_sender_timeout           | 60000   | ms   | t
 wal_writer_delay             | 200     | ms   | t
(9 rows)
```

إنهاء الاتصالات

كل أنظمة إدارة قواعد البيانات الارتباطية رئيسية - بما في ذلك بوستجريسكل- تسمح باتصالات متزامنة و متعددة لقاعدة البيانات حتى يتمكن المستخدمين من إجراء المعاملات. وبسبب هذه المعالجة المتزامنة لقواعد البيانات، قد يكون أداء قاعدة البيانات خلال فترات ذروة المعاملات بطيئا أو أن هناك بعض جلسات المنع. من أجل التعامل مع مثل هذه الحالات، قد تضطر إلى إنهاء بعض الجلسات أو الجلسات المحددة القادمة من مستخدم معين حتى نتمكن من إرجاع أداء قاعدة البيانات مرة أخرى إلى وضعها الطبيعي.

كيف ينجز ذلك...

بوستجريسكل يوفر وظيفة pg_terminate_backend لقتل جلسة معينة. على الرغم من أن وظيفة pg_terminate_backend تعمل على اتصال واحد في وقت واحد، يمكننا تضمين pg_terminate_backendby التفافه حول استعلام select لقتل اتصالات متعددة، استنادا إلى معايير التصفية المحددة في بند WHERE.

لإنهاء كافة الاتصالات من قاعدة بيانات معينة، يمكننا استخدام عملية pg_terminate_backend كما يلي:

```
SELECT pg_terminate_backend(pid) FROM pg_stat_activity
WHERE datname = 'testdb1';
```

لإنهاء جميع الاتصالات لمستخدم معين، يمكننا استخدام pg_terminate_backend على هذا النحو:

```
SELECT pg_terminate_backend(pid) FROM pg_stat_activity
WHERE usename = 'agovil';
```

كيف تعمل...

تتطلب وظيفة pg_terminate_backend عمود رمز المهمة "PID" أو المعرف "ID" الوظيفة كمدخل. يتم الحصول على قيمة رمز المهمة من جدول الكتالوج pg_stat_activity. بمجرد أن يتم تمرير رمز المهمة كمدخل إلى وظيفة pg_terminate_backend ، سيتم تلقائيا إلغاء جميع الاستعلامات قيد التشغيل، وسوف يتم إنهاء اتصال معين يطابق معرف الوظيفة كما هو موجود في pg_stat_activity.

كما أن إنهاء الوصلات الخلفية المنتهية مفيدة أيضا في تحرير الذاكرة من عمليات postgres الخاملة التي لم تحرر لأي سبب من الأسباب وكانت تستهلك موارد نظام.

هناك المزيد...

إذا كان المطلب هو إلغاء استعلامات الجارية وليس إنهاء الجلسات الحالية، فيمكننا استخدام وظيفة pg_cancel_backend لإلغاء جميع الاستعلامات النشطة على اتصال. ولكن مع pg_cancel_backendfunction، يمكننا فقط قتل الاستعلامات الجارية في قاعدة بيانات أو الصادرة من قبل مستخدم معين وليس لديها القدرة على إنهاء الاتصالات.

لإلغاء جميع الاستعلامات الجارية المطبقة على قاعدة بيانات يمكننا استخدام وظيفة pg_cancel_backend كالتالي:

```
SELECT pg_cancel_backend(pid) FROM pg_stat_activity WHERE
datname = 'testdb1';
```

لإلغاء جميع الاستعلامات الجارية الآتية من مستخدم معين؛ فيمكننا استخدام وظيفة pg_cancel_backend كالتالي:

```
SELECT pg_cancel_backend(pid) FROM pg_stat_activity WHERE
usename = 'agovil';
```

في الإصدارات ما قبل بوستجريسكل ٩.٢ كان عمود procpid يمرر كمدخل لـ pg_terminate_backend و pg_cancel_backend لقتل الجالسات الجارية وإلغاء الاستعلامات، حل عمود pid مكان procpid في بوستجريسكل ٩.٢ وما بعدها.

يمكنك الرجوع لروابط التالية لمزيد من المعلومات حول إنهاء الاتصالات الخلفية:

https://blog.sleeplessbeastie.eu/2014/07/23/how-to-terminate-postgresql-sessions/

http://www.devopsderek.com/blog/2012/11/13/list-and-disconnect-postgresql-db-sessions/

٢- السيطرة على الأمن

في هذا الفصل، سنغطي الوصفات التالية:

- تأمين كائنات قاعدة البيانات
- التحكم بالولوج عبر الجدران النارية
- التحكم في النفاذ عبر إعداد الملفات
- اختبار الاتصال عن بعد
- تدقيق تعديلات قاعدة البيانات
- تفعيل " أس أس أل" "SSL" في بوستجريسكل PostgreSQL
- اختبار تشفير " طبقة المقابس الآمنة " "SSL"
- تشفير البيانات السرية
- تكسير كلمات السر بوستجريسكل

مقدمة

تستخدم قواعد البيانات لتخزين البيانات بطريقة منظمة. ويحتفظ بجميع البيانات ذات الصلة بالمؤسسة في قواعد البيانات. وبما أن جميع المعلومات المتعلقة بالشركات مخزنة في قواعد البيانات، يصبح من الضروري وضع ضوابط على الولوج إلى البيانات ولا يسمح إلا للأشخاص المرخص لهم بالنفاذ إلى البيانات ذات الصلة. وفي هذا السياق، يكتسي أمن قاعدة البيانات أهمية قصوى لأنه من المهم ضمان حماية المعلومات المخزنة في قواعد البيانات من المحاولات الخبيثة لعرض وتعديل البيانات من قبل القراصنة أو الأشخاص الذين لديهم نوايا خبيثة.

ويتناول أمن قاعدة البيانات تدابير أمن المعلومات التي تتخذ لحماية قواعد البيانات من أجل ضمان السرية والنزاهة وتوافر البيانات.

يجب حماية قواعد البيانات من مختلف المخاطر والتهديدات، مثل سوء الاستخدام من قبل مستخدمي قاعدة البيانات المرخص لهم، أو المحاولات الخبيثة التي يقوم بها القراصنة لسرقة المعلومات أو إتلاف البيانات، أو إحداث عيوب في التصميم أوخلل في البرمجيات في قواعد البيانات التي تؤدي إلى مختلف الثغرات الأمنية التي يتم استغلالها من قبل القراصنة، و إتلاف البيانات التي قد تكون ناجمة عن إدخال بيانات خاطئة أو أخطاء بشرية، إمكانية تخريب البيانات، وخيار المدير للحفاظ على كلمة مرور افتراضية و التي قد تسمح للأشخاص ذوي النوايا الخبيثة و غير المرخص لهم بالنفاذ إلى البيانات.

تأمين كائنات قاعدة البيانات

من المهم التأكد من أن المستخدمين المصادق عليهم لا يحصلون إلا على البيانات التي يؤذن لهم بالنفاذ إليها. ومع ذلك، يبقى السؤال المتعلق بكيفية إبقاء المستخدمين المصادق عليهم بعيدا عن النفاذ إلى البيانات غير المصرح بها. في بوستجريسكل، يتم تنفيذ ذلك من خلال الحفاظ على سياسة قوية لمراقبة الدخول.

قائمة التحكم بالنفاذ تحكم المستخدمين المسموح لهم بتحديد وتحديث وتعديل الكائنات داخل قاعدة البيانات. وتوضع مجموعة من القيود والضوابط على كل كائن قاعدة بيانات تحديد من يسمح له بالنفاذ إلى هذا الكائن. يتم إعداد حقوق التحكم في النفاذ إلى كائنات قاعدة البيانات من خلال استخدام أوامر "توفير" "GRANT" و "سحب" "REVOKE".

كيف ينجز ذلك...

لا يملك مستخدم قاعدة البيانات عادة حقوق النفاذ إلى أي من كائنات قاعدة البيانات بخلاف تلك التي يملكها. ووفقا لمتطلبات العمل،وتمنح صلاحيات النفاذ إلى كائنات قاعدة البيانات المناسبة للمستخدمين الآخرين من قبل مالك هذه الكائنات. ومع ذلك، إذا جاء متطلب لإلغاء الإذن بعد منح المستخدم حق النفاذ إلى الكائن، إذن يمكن إصدار الأمر "سحب" "REVOKE".

سنناقش حالتين هنا:

- إبطال جميع الأذونات على الجدول لمستخدم معين. هنا، نعرض كيفية الاستخدام:

من الأمر "سحب" "REVOKE"

```
REVOKE ALL on testusers from nchhabra;
```

- إلغاء أذونات محددة على الجدول من مستخدم معين, كما هو موضح هنا:

```
REVOKE insert,update,delete,truncate on testusers from agovil;
```

كيف تعمل...

عادة، يكون لدى جميع المستخدمين مجموعة من الحقوق، والتي تشمل: "تحديد " SELECT " ، "تحديث " UPDATE " ، "حذف " DELETE "، " إدراج " INSERT" ، "ترانكيت" "TRUNCATE " ، و" تريجر" "TRIGGER"، على جميع الجداول التي تم إنشاؤها حديثا من خلال دور"عام" "PUBLIC".

من أجل ضمان أن مستخدم معين لم يعد قادرا على النفاذ إلى الجدول، يجب سحب حقوق هذا الجدول من كلاهما, أي الدور "عام" "PUBLIC" والمستخدم. في المثال السابق قمنا بسحب كافة الأذونات من جدول testusers من المستخدم nchhabra. في هذه الحالة، يتم تقييد المستخدم nchhabra و هذا لمنعه من القيام بأي عملية على جدول testusers .

في السيناريو الثاني من المثال السابق، قمنا بسحب الإذن بإدراج وتحديث وحذف و بتر العمليات على جدول testusers من مستخدم بوستجريسكل، "أغوفيل" "agovil"، مما يسمح للمستخدم بإجراء عمليات القراءة فقط على الجدول عبر بند الجدول "تحديد" "SELECT".

التحكم بالنفاذ عبر الجدران النارية

الطريقة الأساسية لحماية خدمات الشبكة على الخادم هي من خلال جدار ناري. الجدار الناري يمكن أن يكون في شكل كلّ من العتاد والبرمجيات. الجدار الناري يسمح لك بإعداد و تحديد أيٍّ من العملاء المسموح لهم بتمرير الحزم من خلاله لتطبيقات محددة.

الجدار الناري للخادم الباب الرئيسي للنظام على الشبكة. يمكن أن تمنع محاولات النفاذ إلى الخدمات الفردية على الخادم قبل أن تمر الحزم إلى تطبيقات الخادم. هذا بمثابة خط الدفاع الأول في حماية قواعد البيانات بوستجريسكل من الهجوم والتطفل.

كيف ينجز ذلك...

فيما يلي سلسلة من الخطوات المطلوبة لإعداد النفاذ إلى قاعدة البيانات من خلال الجدار الناري على "ريدهات لينوكس" " Red Hat" والتوزيعات الأخرى القائمة على "ريد هات" " Red Hat":

١. لتكون آلات الشبكة قادرة على النفاذ إلى خادم بوستجريسكل ، يجب عليك إعداد قواعد الجدار الناري يدويا للسماح بالنفاذ إلى منذ خادم بوستجريسكل .افتراضيا، يستمع بوستجريسكل إلى "منفذ" "Port" "تي سي بي" "5432" TCP. لذلك، تحتاج إلى تفعيل لـ"منفذ" "5432" Port على الجدار الناري . في بيئات لينكس، يمكنك تفعيل "المنفذ" "5432" Port بواسطة تعديل قواعد "إبي تابلز" "IPTABLES". لهذا، تحتاج إلى فتح الملف الذي يحتوي على قواعد الجدار الناري . هذا الملف يمكن العثور عليه في:

.vim/etc/sysconfig/iptables

بمجرد فتح الملف، تحتاج إلى إضافة القاعدة التالية لتفعيل النفاذ لل«منفذ» "5432" Port:

```
-A INPUT -m state --state NEW -m tcp -p tcp --dport 5432
-j ACCEPT
```

بعد ذلك، تحتاج إلى حفظ التغييرات وإعادة تحميل ملف الإعداد الذي يحتوي على قواعد جدار الحماية من أجل ضمان أن التغييرات الجديدة التي أدخلت هي فعلا حيز التنفيذ، وذلك باستخدام الأمر التالي:

```
service iptables restart
```

في سلسلة الخطوات التالية المطلوبة لإعداد النفاذ إلى "منفذ" "Port" قاعدة البيانات من خلال الجدار الناري على ويندوز ٧. لتفعيل "منفذ" "5432" Port على ويندوز تحتاج إلى اتباع الخطوات التالية بالترتيب:

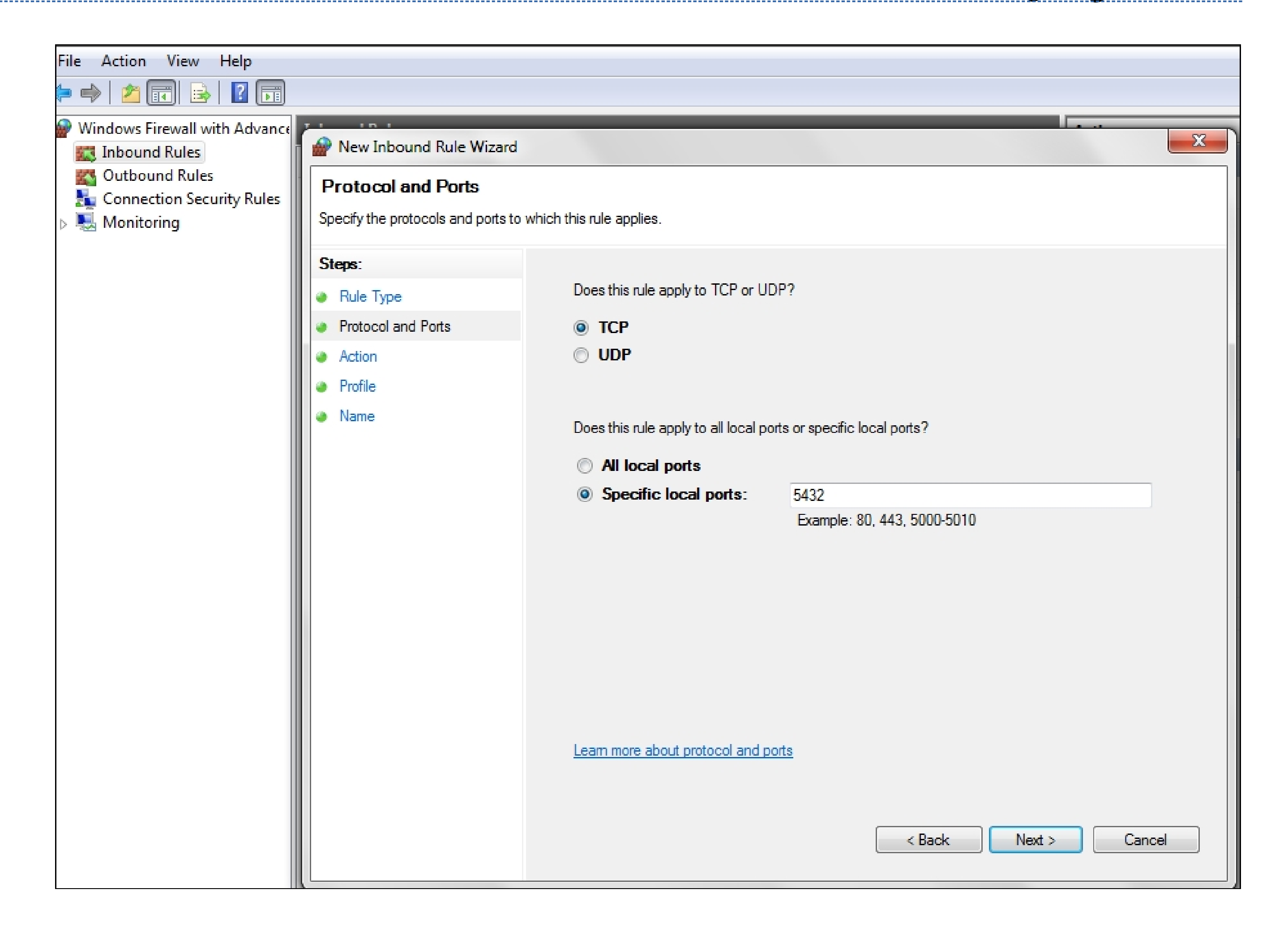
١- افتح الجدار الناري للويندوز من خلال الانتقال إلى **ابدأ | لوحة التحكم | أنظمة والأمن | جدار حماية ويندوز**.

٢- في الجزء الأيمن، انقر على **الإعدادات المتقدمة**. إذا تمت مطالبتك بكلمة مرور المسؤول أو التأكيد، اكتب كلمة المرور أو قم بتقديم التأكيد.

٣- في مربع حوار **الجدار الناري للويندوز مع الحماية المتقدمة** ، انقر فوق **القواعد الواردة** في الجزء الأيمن ثم انقر فوق **قاعدة جديدة** في الجزء الأيسر.

٤- في المربع **مرشد القاعدة الواردة الجديد**، انقر على خيار **"ميثاق" "Protocol"** و **"المنفذ" "Port"** ثم انقر على أزرار الانتقاء، كما هو مبين في الصورة في الأسفل، وأخيرا انقر على زر" **التالي**".

٥- في الشاشة التالية، حدد الإعدادات الافتراضية لجميع الخيارات، أدخل المنفذ رقم ٥٤٣٢ في **المنافذ المحلية المحددة**: حقل النص، ثم الاستمرار في النقر على زر **"التالي"** حتى يظهر لك زر **"الانتهاء"**. أدخل اسما للقاعدة ثم انقر على **إنهاء**.



كيف تعمل...

الخوادم التي تستضيف قواعد بيانات الخاصة بالإنتاج لديها سياسات جدار ناري التي تسمح أو تمنع "منفذ" "Port" أو "عنوان أي بي" " IP ADRESS".

افتراضيا، الجدار الناري يمنع كل شيء. من أجل تفعيل النفاذ إلى التطبيقات، تحتاج إلى إعداد قواعد في سبيل التحكم في النفاذ إلى جدار الحماية للسماح بالنفاذ إلى التطبيق.

سوف تحتاج إلى تفعيل النفاذ على "منفذ" "5432 Port"، وهو المنفذ الافتراضي لـ بوستجريسكل من أجل النفاذ إلى خادم بوستجريسكل

التحكم في النفاذ عبر ملفات الضبط

عندما يتم إعداد الجدار الناري للسماح بالنفاذ إلى خادم بوستجريسكل، تحتاج إلى إعداد خادم بوستجريسكل للسماح بالاتصالات عن بعد. ويتم تنفيذ ذلك من خلال إجراء التغييرات اللازمة في ملفات gg.hf.confconfiguration .postgresql.confand

يحتوي ملف postgresql.conf على إدخال واحد يتحكم في واجهات الشبكة التي تستمع بوستجريسكل للاتصالات.

يستخدم ملف pg_hba.conf لتحديد أي من العملاء يمكنه الاتصال بقاعدة البيانات التي تستخدم دور محدد في تسجيل الدخول.

كيف ينجز ذلك...

١- يلزمك إعداد مُعامِل listen_addresses من أجل تفعيل عملاء الشبكة عن بعد لإجراء اتصال إلى خادم بوستجريسكل:

```
listen_addresses = '*'
```

٢- هنا، يمكنك استخدام علامة النجمة كقيمة لإعداد معامل listen_addresses. هذا الإعداد للمعامل يقوم بتفعيل جميع منافذ "Ports" الشبكة.

٣- الخطوة التالية هي إجراء تغييرات في إعداد ملف pg_hba.conf. هذه التغييرات تحدد قواعد النفاذ في سبيل السماح للاتصالات عن بعد بالنفاذ إلى خادم بوستجريسكل.

٤- افتح ملف pg_hba.conf تحت دليل البيانات أو ضمن الدليل المحدد بواسطة بيئة المتغير $PGDATA وتحديد قواعد التحكم بالنفاذ الضرورية:

```
# TYPE    DATABASE  USER   CIDR-ADDRESS       METHOD        OPTION
host      hrdb      all    192.168.12.10/32   md5
host      all       all    192.168.54.1/32    reject
host      all       all    192.168.1.0/24     trust
host      hrd       all    192.168.1.10/24    crypt
```

٥- الإدخال الأول في ملف pg_hba.conf يدل على أنه يسمح لأي مستخدم من المضيف ١٩٢.١٦٨.١٢.١٠ للاتصال بقاعدة البيانات "hrdb" إذا تم توفير كلمة مرور المستخدم بشكل صحيح.

٦- يظهر الإدخال الثاني في ملف pg_hba.conf يعرض سجل المضيف الذي سيتم رفض كافة المستخدمين من المضيف ١٩٢.١٦٨.٥٤.١ لأي من قواعد البيانات المطلوبة.

٧- يظهر الإدخال الثالث في ملف pg_hba.conf سجل مضيف يسمح لأي جهاز على الشبكة الفرعية ١٩٢.١٦٨.١.٠ بالاتصال و النفاذ إلى أي قاعدة بيانات دون تحديد أي كلمة مرور في الأساس، مع طريقة "ترست" "trust"، ونحن نعتمد على استيثاق المضيف في استخدام هذه الطريقة.

٨- الإدخال الأخير في ملف pg_hba.conf ينص على أنه يسمح لأي مستخدم لديه "عنوان أيبي" ١٩٢.١٦٨.١.١٠ ومع كلمة مرور صالحة بالاتصال بقاعدة البيانات "hrdb".

ومع ذلك، هنا سيتم تشفير كلمة المرور أثناء الاستيثاق بسبب مصطلح "كريبت" " crypt" ، الذي تم تحديده كطريقة للاستيثاق.

كيف تعمل...

يتحكم في استيثاق العميل بواسطة إعداد ملف pg_hba.conf. تقوم الإدخالات في ملف pg_hba.conf بإعداد أذونات الاستيثاق والتخويل للمضيف.

ستقرأ مدخلات ملف pg_hba.conf للاستيثاق كلما يتم تلقي طلب اتصال. في البداية، يستخدم ملف pg_hba.conf لتحديد ما إذا كان العميل لديه الإذن بالاتصال بكائنات قاعدة البيانات أو لا.

. بمجرد أن يتم تحديد أنه يسمح للمستخدم للنفاذ إلى قاعدة البيانات، فإن الخطوة التالية ستكون التأكد من استيفاء العميل لجميع الشروط للاستيثاق عليه بنجاح.

حتى ولو تم الاستيثاق على المستخدم وأن له أذونات للاتصال بقاعدة البيانات، فإن أيا من الأذونات على مستوى الجدول سوف تبقى مطبقة على قاعدة البيانات. يمكنك التحقق من الأذونات على قاعدة البيانات باستخدام مبدل z\ ، كما هو موضح في الصورة أدناه:

```
hrdb=# \z
                         Access privileges
 Schema | Name | Type  |    Access privileges     | Column access privileges
--------+------+-------+--------------------------+--------------------------
 public | test | table | postgres=arwdDxt/postgres+|
        |      |       | agovil=arwdD/postgres    +|
        |      |       | nchabbra=r/postgres       |
 public | x    | table | postgres=arwdDxt/postgres+|
        |      |       | agovil=arwdDxt/postgres  +|
        |      |       | nchabbra=r/postgres       |
(2 rows)
```

أثناء تضبيط اتصال قاعدة البيانات، تقرأ الإدخالات في ملف pg_hba.conf من أعلى إلى أسفل. لحظة العثور على إدخال متطابق، سوف يقوم بوستجريسكل بإيقاف البحث، وسوف يتم السماح أو رفض الاتصال على أساس القواعد المذكورة و الخاصة بالإدخال الذي وجد. سوف يفشل الاتصال تماما إذا لم يكن الإدخال متطابق في ملف pg_hba.conf.

هناك المزيد...

يعرّف نوع طريقة الاستيثاق المعروف باسم "الهوية" " ident " في ملف الإعداد pg_hba.conf. تعمل طريقة استيثاق "الهوية" " ident " من خلال الحصول على اسم مستخدم نظام التشغيل الخاص بالعميل واستخدامه كاسم مستخدم لقاعدة البيانات المسموح به.

إذا استخدمت طريقة استيثاق "الهوية" لإدخال المضيف في ملف pg_hba.conf، إذن يجب تحديد خريطة "الهوية" أو المسماة "تعيين" " mapping". يحدد هذا الخيار في ملف الإعداد pg_ident.conf، ويستخدم لتعيين تعريف اسم المستخدم ، وهذا هو اسم المستخدم لنظام التشغيل العميل مع مستخدم قاعدة البيانات "PostgreSQL" الموجود مسبقا.

على غرار ملف pg_hba.conf، يوجد الملف pg_ident.conf أيضا في دليل البيانات أو في المسار المحدد بواسطة متغير بيئة "PGDATA".

أولا، يجب تعيين عبارة "ident " كطريقة للاستيثاق في ملف pg_hba.conf، كما يلي:

```
# TYPE     DATABASE    USER        CIDR-ADDRESS          METHOD      OPTION
  host     hrdb        all         192.168.12.10/32 ident        hruser
```

هنا، في ملف pga_hba.conf، يمكن لأي مستخدم يستخدم "عنوان أيبي" ١٩٢.١٦٨.١٢.١٠ الاتصال بقاعدة البيانات "hrdb" باستخدام "hruser " "mapname"، الذي هو في الأساس تعيين لأسماء المستخدمين " يونكس" "UNIX" و اسم مستخدم قاعدة البيانات المقابلة " PostgreSQL". يتم تحديد هذه الإدخالات في ملف pg_ident.conf، كما يلي:

```
# MAPNAME         Ident-USERNAME          PG-USERNAME
    hruser        govil_amit              agovil
    hruser        kumar_neeraj            agovil
```

تم ضبط خارطة الهوية " hruser" الآن في ملف pg_ident.conf. تسمح الإدخالات في ملف pg_ident.conf لأي من مستخدمي نظام "UNIX" و govil_amit و kumar_neeraj بالاتصال بقاعدة بيانات " hrdb" باستخدام حساب مستخدم نظام " agovil" "PostgreSQL".

لمزيد من المعلومات حول الإدخالات في ملف pg_hba.conf، يمكنك الرجوع إلى http://www.postgresql.org/docs/9.3/static/auth-pg-hba-conf.html.

اختبار الاتصال عن بعد

بعد تضبيط بيئة الشبكة في " PostgreSQL"، عادة ما تكون فكرة جيدة لاختبارها.

كيف ينجز ذلك...

يمكنك استخدام برنامج "psql " لاختبار الاتصالات بخادم " PostgreSQL" من

عميل عن بعد:

```
D:\Postgresql_Project\bin>psql -h 192.168.12.10 hrdb agovil
Password for user agovil:
psql (9.3.4)
WARNING: Console code page (437) differs from Windows code page (1252)
8-bit characters might not work correctly. See psql reference
page "Notes for Windows users" for details.
Type "help" for help.
Hrdb=>
```

كيف تعمل...

بعد تفعيل استيثاق العميل بين خادم " PostgreSQL" وتطبيق العميل، وكذلك بعد إعداد قواعد التحكم بالنفاذ في ملف الإعداد pg_hba.conf الخاصة بالمضيف ، فمن الجيد اختبار الاتصال عن بعد.

سيساعدك هذا على معرفة ما إذا كان أعدت قواعد التحكم بالنفاذ بشكل صحيح في ملف pg_hba.conf وعما إذا كان العملاء يواجهون أية أخطاء اتصال على الرغم من السماح لهم النفاذ استنادا إلى قواعد ملف إعداد المضيف.

تدقيق تعديلات قاعدة البيانات

يبقى أمن قاعدة البيانات مصدر قلق لأي تطبيق قاعدة بيانات. ولأغراض التدقيق، من المهم تحديد البيانات التي تم تعديلها، ومن التي أجرت هذا التعديل، ومتى وكيف تم تنفيذ هذا التعديل في بيئة الإنتاج.

يمكن استخدام مشغل سجل التعديل كآلية لتحديد التعديلات التي تجرى على البيانات في قاعدة بيانات " PostgreSQL" والإجابة على جميع الأسئلة ذات الصلة من منظور التدقيق.

كيف ينجز ذلك...

١- أولا، إنشاء مخطط، ثم تخزين الكائنات الأخرى المطلوبة لتعقب التعديلات في هذا المخطط.

يمكنك إنشاء المخطط كما يلي:

```
CREATE SCHEMA logging;
```

٢- ستكون الخطوة التالية هي إنشاء جدول لتخزين بعض السجلات لتتبع التعديلات ، كما يلي:

```
CREATE TABLE logging.t_history (
id serial,
tstamp timestamp DEFAULT now(),
schemaname text,
tabname text,
operation text,
who text DEFAULT current_user,
new_val json,
old_val json
);
```

تتمثل نقطة استخدام هذا الجدول في تتبع جميع التعديلات التي أجريت على الجدول. نريد أن نعرف أي العمليات قد أجريت. المسألة الهامة التالية هي عندما يضاف صف جديد، سوف يكون مرئيا من قبل إجراء " trigger".وهذا نفس الشيء بخصوص حذف التعديلات.

٣- بعد ذلك، قم بإنشاء دالة تقوم بتسجيل التعديلات ، بما في ذلك التعديلات القديمة والقيم في الجدول t_history. عرفت الدالة بطريقة تسمح بتعقب كافة عمليات " DML" -بما في ذلك الإدخالات"inserts " " والتحديثات "Updates" والمحذوفات "Deletes"- وبناء على نوع عمليات " DML"، تقوم بتسجيل البيانات - بما في ذلك التعديلات - في جدول t_history:

```
CREATE FUNCTION change_trigger()
RETURNS trigger AS $$
BEGIN
 IF TG_OP = 'INSERT' THEN
   INSERT INTO logging.t_history (
   tabname,
   schemaname,
   operation,
   new_val
   )
   VALUES (
   TG_RELNAME,
   TG_TABLE_SCHEMA,
   TG_OP,
   row_to_json(NEW)
  );
 RETURN NEW;
 ELSIF TG_OP = 'UPDATE' THEN
   INSERT INTO logging.t_history (
```

```
    tabname,
    schemaname,
    operation,
    new_val,
    old_val
     )
    VALUES (
    TG_RELNAME,
    TG_TABLE_SCHEMA,
    TG_OP,
    row_to_json(NEW),
    row_to_json(OLD)
 );
 RETURN NEW;
 ELSIF TG_OP = 'DELETE' THEN
    INSERT INTO logging.t_history (
    tabname,
    schemaname,
    operation,
    old_val
  )
 VALUES (
    TG_RELNAME,
    TG_TABLE_SCHEMA,
    TG_OP,
    row_to_json(OLD)
 );
 RETURN OLD;
 END IF;
END;
$$ LANGUAGE 'plpgsql' SECURITY DEFINER;
```

٤- الآن، إنشاء جدول مع بعض البيانات فيه واستخدام هذا الجدول لإجراء تعديلات:

```
CREATE TABLE t_trig (id int,name text);
```

٥- الخطوة التالية هي إنشاء مشغل سجل التعديل الذي سينفذ قبل حدوث أي حدث " DML"على الجدول t_trig، الذي تم إنشاؤه في الخطوة السابقة:

```
CREATE TRIGGER t BEFORE INSERT OR UPDATE OR DELETE ON
t_trig FOR EACH ROW EXECUTE PROCEDURE change_trigger();
```

٦- الآن، إجراء تعديلات في الجدول t_trig واختبار تنفيذ " trigger " ، على النحو التالي:

```
INSERT INTO t_trig VALUES (1, 'hans');
UPDATE t_trig SET id = 10 * id, name ='paul';
```

٧- بعد ذلك، تحقق مما إذا كانت جداول السجل تحتوي على تعديلات أجريت على الجداول الأساسية باستخدام الاستعلام، كما هو موضح في الصورة أدناه:

```
postgres=# select tabname, operation,old_val ,new_val  from logging.t_history ;
 tabname | operation |        old_val         |          new_val
---------+-----------+------------------------+--------------------------
 t_trig  | INSERT    |                        | {"id":1,"name":"hans"}
 t_trig  | UPDATE    | {"id":1,"name":"hans"} | {"id":10,"name":"paul"}
(2 rows)
```

كيف تعمل...

يمكن استخدام وظيفة " trigger " العامة لتسجيل التعديلات في جدول التاريخ. حيث تسجل السجلات القديمة والجديدة والجداول المتضررة، والمستخدمين الذين قاموا بالتعديل، ونوع عملية " DML"، والطابع الزمني لكل تعديل.

من المهم التأكد من أن السجل، الذي يحتفظ به للحفاظ على مسار التعديلات ، لا يمكن تغييره من قبل أي شخص مرخص له. ويمكن ضمان هذا عن طريق وضع علامة على وظيفة " trigger " كمحدد الأمن "SECURITY DEFINER " . سيضمن ذلك عدم تنفيذ الوظيفة " function" نفسها من قبل المستخدم الذي يقوم بالتعديل ولكن من قبل المستخدم الذي كتب الوظيفة " function".

ولا يمكن استخدام هذا النوع من التلقائية القائمة على " trigger " لتتبع الأنشطة التالية من منظور مراجعة الحسابات:

- -لا يمكنها تدقيق عبارات "SELECT" .
- -لا يمكنها تدقيق جداول النظام.
- - لا يمكنها تدقيق عمليات " DDL" كعبارة جدول التغيير " ALTER TABLE ".

هناك المزيد...

هناك طريقة أخرى لجمع تعديلات البيانات التي أدخلت على " PostgreSQL". تتضمن هذه التعديلات أيضا التعديلات التي أجريت على بيانات " DDL". يمكننا جمع التعديلات التي أجريت على نظام " PostgreSQL"من سجل ملف الخادم.

من أجل جمع تعديلات البيانات من سجل الخادم ، تحتاج إلى تعديل إعداد log_statement وتعيين قيمته إما " mod" أو "الكل" "All" في ملف الإعداد postgresql.conf.

وبمجرد الانتهاء من ذلك، تحتاج إلى إعادة تحميل الإعداد على النحو التالي:

```
pg_ctl -D /var/lib/pgsql/data reload
```

تفعيل SSL في بوستجريسكل

بشكل افتراضي، يعد خادم " PostgreSQL" لقبول اتصالات العميل البعيد باستخدام اتصال "TCP " قياسي. المسألة مع هذا النوع من اتصالات الشبكة هو أن إرسال البيانات يكون في نص واضح عبر الشبكة، ويصبح طبعا خاضع للمراقبة . حيث يستطيع أي شخص يستخدم برنامج مراقبة "Sniffer" أن يعترض البيانات المرسلة في نص واضح، وبهذه الطريقة، يمكن أن تتعرض سرية البيانات للخطر.

الآن، السؤال المناسب هو: ما هي البيانات في " PostgreSQL" التي تكون قابلة للاعتراض. إن عبارة "SQL" المرسلة بواسطة الأداة المساعدة "PSQL" إلى الخادم ومجموعة النتائج التي أنشئت بواسطة خادم " PostgreSQL" هي بعض من الأشياء التي هي قابلة للاعتراض. إن تمكين المعترض لرؤية نتيجة الاستعلامات يعني أنه قادر على مشاهدة بيانات الجداول الخاص بك.

للتعامل مع هذه الحالة، يدعم " PostgreSQL" طبقة المقابس الآمنة " SSL" جلسات "TCP " المشفرة. تستخدم جلسات تشفير "TCP " المستندة إلى طبقة المقابس الآمنة " SSL" مفتاح التشفير لتشفير البيانات قبل إرسالها على الشبكة. خادم "PostgreSQL" وجهاز العميل يمرران مفتاح التشفير الذي يستخدم لتشفير البيانات.

كيف ينجز ذلك...

من أجل التعامل مع هذا، تحتاج إلى تفعيل دعم " SSL" في "PostgreSQL". يمكن القيام بذلك عن طريق تعديل قيمة إعداد معامل " SSL" في ملف postgresql.conf:

```
ssl = on
```

عند إعادة تشغيل خادم PostgreSQL، فإنه سيتعرف على التعديل في الإعداد وتفعيل اتصالات SSL. سوف يستمع خادم PostgreSQL الآن لكل من اتصالات TCP العادية، وكذلك اتصالات TCP المستندة إلى طبقة المقابس الآمنة "SSL" على نفس المنفذ.

ومع ذلك، بمجرد تفعيل SSL ، فإن خادم بوستجريسكل يقوم بالتأكد من أن مفاتيح التشفير أو ملفات الشهادة متوفرة في دليل بيانات بوستجريسكل، وإلا فإنه لن يبدأ حتى يجدهم.

كيف تعمل...

الآن بعد تفعيل دعم SSL في بوستجريسكل، لدعم جلسة SSL ، يجب أن يكون خادم بوستجريسكل قادرا على النفاذ إلى كل من مفتاح التشفير والشهادة. يستخدم بروتوكول SSL مفتاح التشفير لتشفير بيانات الشبكة، بينما يستخدم العميل البعيد الشهادة المقدمة من قبل الخادم للتحقق من أن مفتاح التشفير يأتي من مصدر موثوق به.

ينشئ مفتاح التشفير من شهادة موقعة من قبل مؤسسة موثوقة. ويمكن استخدام هاتين الطريقتين للحصول على شهادة:

- يمكنك شراء شهادة من سلطة استيثاق مثل Verisign أو Thawte .

- في حالات أخرى، يمكنك إنشاء شهادة موقعة ذاتيا، تشير إلى أن التشفير جاء من طرفك.

هناك المزيد...

نحن هنا بصدد إنشاء شهادة موقعة ذاتيا ومفاتيح التشفير باستخدام أداة مفتوحة المصدر تعرف باسم " OpenSSL". بالنسبة لنظام التشغيل ويندوز، تحتاج إلى تنزيل أحدث إصدار من حزمة Win32OpenSSL.

الخطوات التالية مطلوبة لإنشاء مفتاح التشفير وملفات الشهادة الموقعة ذاتيا:

١- إنشاء مفتاح تشفير حماية عبارة المرور.

٢- قم بإزالة عبارة المرور الرئيسية

٣- إنشاء الشهادة الموقعة ذاتيا

الخطوة الأولى هي إنشاء مفتاح التشفير المستخدم من قبل بوستجريسكل لتشفير جلسات SSL. يتم ذلك باستخدام الخيار req OpenSSL :

```
C:\cygwin\OpenSSL-Win32\bin>openssl req -new -text -out server.req
Loading 'screen' into random state - done
Generating a 1024 bit RSA private key
.......................++++++
..........................................................++++++
writing new private key to 'privkey.pem'
Enter PEM pass phrase:
Verifying - Enter PEM pass phrase:
-----
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [AU]:IN
State or Province Name (full name) [Some-State]:CHANDIGARH
Locality Name (eg, city) []:CHANDIGARH
Organization Name (eg, company) [Internet Widgits Pty Ltd]:
Organizational Unit Name (eg, section) []:
Common Name (e.g. server FQDN or YOUR name) []:
Email Address []:chitij24@hushmail.com

Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:
An optional company name []:

C:\cygwin\OpenSSL-Win32\bin>
```

الخطوة السابقة كانت لإنشاء ملف يسمى privkey.pem الذي يحتوي على مفتاح التشفير وملف server.req الذي يحتوي على شهادة أساسية:

```
C:\cygwin\OpenSSL-Win32\bin>dir server.req privkey.pem
 Volume in drive C has no label.
 Volume Serial Number is 5212-4294

 Directory of C:\cygwin\OpenSSL-Win32\bin

06/16/2014  12:56 PM             2,164 server.req

 Directory of C:\cygwin\OpenSSL-Win32\bin

06/16/2014  12:56 PM             1,041 privkey.pem
               2 File(s)          3,205 bytes
               0 Dir(s)   3,397,812,224 bytes free
```

كما ذكر سابقا، الملف الذي يحتوي على مفتاح تشفير privkey.pem محمي بواسطة عبارة مرور. يمكنك إزالة عبارة المرور هذه من مفتاح التشفير باستخدام خيار SSL آخر، كما يلي:

```
C:\cygwin\OpenSSL-Win32\bin>openssl rsa -in privkey.pem -out server.key
Enter pass phrase for privkey.pem:
writing RSA key

C:\cygwin\OpenSSL-Win32\bin>dir server.key
 Volume in drive C has no label.
 Volume Serial Number is 5212-4294

 Directory of C:\cygwin\OpenSSL-Win32\bin

06/16/2014  01:12 PM               887 server.key
               1 File(s)            887 bytes
               0 Dir(s)   3,398,086,656 bytes free
```

عندما يتم تنفيذ الأمر OpenSSL المذكور سابقا،يقوم OpenSSL بالسؤال عن عبارة المرور لمفتاح التشفير. ثم يقوم بإنشاء مفتاح تشفير جديد يسمى server.key، والذي لا يتطلب إدخال عبارة المرور.

الآن بعد أن يكون لديك مفتاح تشفير بدون عبارة مرور وشهادة أساسية، فإن الخطوة التالية هي تحويل الشهادة إلى تنسيق X.509 القياسي والتوقيع عليه باستخدام مفتاح التشفير:

```
C:\cygwin\OpenSSL-Win32\bin>openssl req -x509 -in server.req
-text -key server.key -out server.crt
Loading 'screen' into random state — done
```

يتم إنشاء الشهادة في وضع النص باستخدام تنسيق X.509 القياسي وحفظها في الملف server.crt. والآن أنشئت الشهادة في وضع النص، فلنلق نظرة عليها:

```
C:\cygwin\OpenSSL-Win32\bin>type server.crt | more
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number:
            94:be:08:1d:22:b5:58:b0
    Signature Algorithm: sha1WithRSAEncryption
        Issuer: C=IN, ST=CHANDIGARH, L=CHANDIGARH, O=Internet Widgits Pty Ltd/em
ailAddress=chitij24@hushmail.com
        Validity
            Not Before: Jun 16 07:50:29 2014 GMT
            Not After : Jul 16 07:50:29 2014 GMT
        Subject: C=IN, ST=CHANDIGARH, L=CHANDIGARH, O=Internet Widgits Pty Ltd/e
mailAddress=chitij24@hushmail.com
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
                Public-Key: (1024 bit)
                Modulus:
                    00:e1:06:75:9d:c4:3b:93:24:50:77:41:82:f1:ed:
                    de:f3:2c:b0:1e:3a:3c:69:d5:8b:1b:38:9a:6c:ac:
                    2e:94:ab:20:a0:3d:38:7b:ea:32:7d:8a:48:70:1f:
                    13:23:5c:1e:a0:ef:0e:b4:25:05:94:52:7c:9d:a9:
                    e2:88:18:17:9b:ef:89:91:1d:ca:49:7d:34:b7:f7:
                    ea:ad:ad:f8:b9:68:e0:2d:14:ec:e0:42:d1:fd:4e:
                    93:fc:a2:d2:72:12:de:87:07:02:7f:94:ab:e5:de:
                    b0:df:ab:03:a2:a1:3f:a4:eb:14:52:15:f1:7a:d8:
                    e5:cd:82:df:25:e8:fa:b6:2b
                Exponent: 65537 (0x10001)
        X509v3 extensions:
            X509v3 Subject Key Identifier:
                C8:4F:45:8B:45:D6:4C:31:61:B5:19:34:1E:32:8E:21:51:5B:9C:3E
            X509v3 Authority Key Identifier:
                keyid:C8:4F:45:8B:45:D6:4C:31:61:B5:19:34:1E:32:8E:21:51:5B:9C:3
E

            X509v3 Basic Constraints:
                CA:TRUE
    Signature Algorithm: sha1WithRSAEncryption
         4f:8a:6f:e6:4d:f9:78:9a:7e:e7:3d:a6:19:27:fb:62:96:1d:
         03:64:55:d5:c7:b1:09:fb:b0:d6:3f:cb:23:88:64:07:54:f7:
         0e:5f:41:92:ae:6c:3b:0f:ec:c4:d7:fe:97:b8:83:ef:c4:0b:
         8d:96:e7:a6:cb:e0:61:38:b5:21:fa:ca:16:ee:6e:ec:d5:02:
         87:28:cd:58:28:78:7b:dc:56:e5:7c:53:8b:a5:a6:9b:72:04:
         79:f0:61:90:ad:d2:eb:0c:46:82:58:99:30:a7:a3:95:9e:d6:
         ad:72:ed:65:c4:1b:78:83:91:f9:f7:ec:3d:96:4e:a6:57:0b:
         92:6b
-----BEGIN CERTIFICATE-----
MIIC0DCCAjmgAwIBAgIJAJS+CB0itViwMA0GCSqGSIb3DQEBBQUAMIGAMQswCQYD
VQQGEwJJTjETMBEGA1UECAwKQ0hBTkRJR0FSSDETMBEGA1UEBwwKQ0hBTkRJR0FS
SDEhMB8GA1UECgwYSW50ZXJuZXQgV2lkZ2l0cyBQdHkgTHRkMSQwIgYJKoZIhvcN
AQkBFhVjaGl0aWoyNEBodXNobWFpbC5jb20wHhcNMTQwNjE2MDc1MDI5WhcNMTQw
NzE2MDc1MDI5WjCBgDELMAkGA1UEBhMCSU4xEzARBgNVBAgMCkNIQU5ESUdBUkgx
EzARBgNVBAcMCkNIQU5ESUdBUkgxITAfBgNVBAoMGEludGVybmV0IFdpZGdpdHMg
UHR5IEx0ZDEkMCIGCSqGSIb3DQEJARYVY2hpdGlqMjRAaHVzaG1haWwuY29tMIGf
MA0GCSqGSIb3DQEBAQUAA4GNADCBiQKBgQDhBnWdxDuTJFB3QYLx7d7zLLAeOjxp
1YsbOJpsrC6UgyCgPTh76jJ9ikhwHxMjXB6g7w60JQWUUnydgeKIGBeb74mRHcpJ
```

الخطوة التالية ستكون نسخ ملفات "server.key " و server.crt إلى دليل بيانات بوستجريسكل ثم قم بإعادة تشغيل خدمة خادم بوستجريسكل.

لمزيد من المعلومات حول سل في بوستجريسكل، يمكنك الرجوع إلى www.postgresql.org/docs/9.3/static/ssl-tcp.html.

اختبار تشفير SSL

عادة، يرسل الاتصال بين تطبيق العميل، مثل psql ، وخادم قاعدة البيانات، وهو بوستجريسكل، عبر الشبكة كنص واضح ويكون عرضة للتنصت. لمنع أي نوع من التنصت أو الاعتراض ، تأكد من أن الاتصال بين العميل psql وخادم قاعدة البيانات مشفر. أنشأنا SSL وقمنا بتفعيل التشفير في الوصفة السابقة. بمجرد نقل ملفات تشفير الشهادة والتشفير إلى دليل البيانات، فإن الخطوة التالية هي التحقق من تشفير SSL. الآن حان الوقت لاختبار تشفير SSL الذي وضعناه في الوصفة السابقة.

كيف ينجز ذلك...

بعد إعادة تشغيل خادم بوستجريسكل، فإن الخطوة التالية هي استخدام تطبيق psql من أجل اختبار اتصال SSL ، كما هو موضح هنا:

```
D:\Postgresql_Project\bin>psql -h 10.168.192.16 hrdb postgres
Password for user postgres:
psql (9.3.4)
WARNING: Console code page (437) differs from Windows code page (1252)
         8-bit characters might not work correctly. See psql reference
         page "Notes for Windows users" for details.
SSL connection (cipher: DHE-RSA-AES256-SHA, bits: 256)
Type "help" for help.

hrdb=#
```

كيف تعمل...

يحاول تطبيق psql الاتصال بخادم بوستجريسكل في وضع SSL أولا وإذا فشل وضع SSL يحاول الاتصال في وضع النص العادي. أثناء إجراء الاتصال إلى العميل psql، يمكنك أن ترى معلومات لافتة تحتوي على كلمات رئيسية مثل اتصال SSL وبعض النصوص المشفرة. يمكن رؤية ذلك في الصورة في الأعلى.

تشفير البيانات السرية

من المهم حماية المعلومات السرية المخزنة في قواعد البيانات، مثل معلومات بطاقة الائتمان، ومعلومات عن المعاملات المالية، والمعلومات الشخصية للموظف. ولا تكفي آليات قواعد البيانات المعتادة، مثل الاحتفاظ بقوائم مراقبة الدخول لتنفيذ ضوابط أمنية مشددة على المعلومات السرية، لضمان عدم وقوع هذه المعلومات الحساسة في أيدي المستخدمين الخبيثين. المهم هو ضمان حفظ البيانات السرية في شكل غير مفهوم للمستخدمين غير المصرح لهم. أما بالنسبة للمستخدمين المخولين، فيجب تحويل المعلومات إلى نسقها الأصلي بحيث يكون مفهوما. هذا هو المكان الذي يظهر فيه

التشفير في الواجهة. التشفير هو عملية تحويل البيانات إلى تنسيق يجعل البيانات غير مقروءة أو غير ملموسة للمستخدمين غير المصرح لهم.

التشفير يترجم البيانات إلى نص الشفرات أو وضع سري، وبالتي لا يمكن فك شفرتها وتحويلها إلى شكلها الأصلي إلا بمساعدة من المفتاح الذي يحتفظ به الأفراد المصرح لهم فقط. لهذا السبب، يعتبر التشفير واحدا من أكثر الطرق فعالية لتحقيق أمن البيانات.

هناك نوعان من التشفير: واحد هو **التشفير المتماثل Symmetric encryption** ، والآخر هو **التشفير غير المتماثل asymmetric system**. يستخدم التشفير المتماثل مفتاح متطابق لكل تشفير وفك تشفير البيانات. خوارزميات مفتاح متماثل حسابيا هي أسرع بكثير من الخوارزميات غير المتماثلة، حيث أن عملية التشفير تكون أقل تعقيدا وتستغرق وقتا أقل. التشفير اللامتماثل يستخدم اثنين من مفاتيح ذات الصلة (العامة والخاصة) لتشفير البيانات وفك التشفير ويزيل المخاطر الأمنية من تبادل المفاتيح. المفتاح الخاص لا يعرض مطلقا. لا يمكن فك تشفير الرسالة التي يتم تشفيرها باستخدام المفتاح العمومي إلا من خلال تطبيق نفس الخوارزمية واستخدام مفتاح خاص مطابق. وبالمثل، لا يمكن فك تشفير الرسالة التي يتم تشفيرها باستخدام المفتاح الخاص إلا باستخدام المفتاح العام المتطابق.

كيف ينجز ذلك...

بوستجريسكل لديها مستويات مختلفة من التشفير للاختيار من بينها. بوستجريسكل يوفر وحدة pgcrypto ، الذي يوفر وظائف التشفير لـ بوستجريسكل.

في القسم التالي، سنقوم بإنشاء جدول واستخدام **معيار التشفير المتقدم** " AES " لتشفير بيانات الجدول ثم فك تشفير البيانات عبر الوظيفتين تشفير" encrypt " وفك تشفير " decrypt":

```
testdb1=# create extension pgcrypto;
testdb1=# create table demo(pw bytea);
testdb1=# insert into demo(pw) values ( encrypt( 'champion', 'key', 'aes') );

testdb1=# select * from demo;
               pw
------------------------------------
\xdf5fa25e36fd16c9e4688bcf46bf11c3
(1 row)
testdb1=# select decrypt(pw, 'key', 'aes') FROM demo;
 decrypt
--------------------
\x6368616d70696f6e
(1 row)
```

```
testdb1=# select convert_from(decrypt(pw, 'key', 'aes'), 'utf-8') FROM demo;
convert_from
--------------
champion
(1 row)
```

كيف تعمل...

وحدة pgcrypto وحدة نمطية في بوستجريسكل وهي توفر التشفير في شكل وظائف قاعدة البيانات. فهي عميل مستقل. وحدة pgcrypto توفر الدعم للتشفير الخام، " بي جي بي" "PGP"Pretty Good Privacy" متوافقة مع التشفير، والتجزئة.

يوفر بوستجريسكل الدعم لكل من التشفير المتماثل وغير المتماثل. للحصول على تشفير أقوى، يمكنك استخدام طريقة التشفير القائم على " PGP" حيث لديك زوجان من المفاتيح: المفتاح العام والخاص، وفي هذه الحالة، يستخدم المفتاح العام لتشفير البيانات ويستخدم المفتاح الخاص لفك تشفير البيانات.

سنقوم الآن باستعراض كيفية استخدام المفتاح العام والخاص لتشفير وفك تشفير البيانات. سنقوم باستخدام الوظائف الأربع التالية للتمثل، جنبا إلى جنب مع شرح لاستخدام هذه الوظائف:

- pgp_pub_encrypt: هذه هي الدالة التي سوف تستخدم لتشفير البيانات الخاصة بك باستخدام المفتاح العام الخاص بك.

- pgp_pub_decrypt : هذه هي الدالة التي سوف تستخدم لفك تشفير البيانات الخاصة بك باستخدام المفتاح الخاص الخاص بك.

- dearmor : تستخدم الدالة dearmor لإخفاء البيانات الثنائية في شكلها الأصلي، وهذا هو، تنسيق درع PGP ASCII ، مما يجعلها مناسبة لتمريرها إلى تشفير.

- pgp_key_id : تستخدم الوظيفة pgp_key_id لاستخراج معرف المفتاح للمفتاح العام أو الخاص. هذه الوظيفة تخبرك بالمفتاح الذي استخدم لتشفير رسالة معينة، بحيث من بين مجموعة من المفاتيح المتاحة، يمكنك استخدام المفتاح المناسب لفك تشفير الرسالة.

فيما يلي سلسلة من الخطوات التي تستخدم لإثبات استخدام كل من المفاتيح العامة والخاصة لتشفير وفك تشفير البيانات باستخدام الوظائف المذكورة سابقا:

١. أولا، إنشاء الجدول الذي سيتم تخزين البيانات فيه:

```
CREATE TABLE testuserscards(
card_id SERIAL PRIMARY KEY,
username varchar(100),
cc bytea
);
```

٢. بعد ذلك، أدخل السجلات في الجدول وقم بتشفير البيانات:

```
INSERT INTO testuserscards(username, cc)
SELECT robotccs.username, pgp_pub_encrypt(robotccs.cc, keys.pubkey) As cc
FROM (VALUES ('robby', '4111111111111111'),
('artoo', '4111111111111112') ) As robotccs(username, cc)
CROSS JOIN (SELECT dearmor('
-----BEGIN PGP PUBLIC KEY BLOCK-----
Version: GnuPG v1.4.1 (GNU/Linux)
mQGiBELIIgoRBAC1onBpxKYgDvrgCaUWPY34947X3ogxGOfCN0p6Eqrx+2PUhm4n
vFvmczpMT4iDc0mUO+iwnwsEkXQI1eC99g8c0jnZAvzJZ5miAHL8hukMAMfDkYke
5aVvcPPc8uPDlItpszGmH0rM0V9TIt/i9QEXetpyNWhk4jj5qnohYhLeZwCgkOdO
RFAdNi4vfFPivvtAp2ffjU8D/R3x/UJCvkzi7i9rQHGo313xxmQu5BuqIjANBUij
8IE7LRPI/Qhg2hYy3sTJwImDi7VkS+fuvNVk0d6MTWplAXYU96bn12JaD21R9sKl
Fzcc+0iZI1wYA1PczisUkoTISE+dQFUsoGHfpDLhoBuesXQrhBavI8t8VPd+nkdt
J+oKA/9iRQ87FzxdYTkh2drrv69FZHc3Frsjw9nPcBq/voAvXH0MRilqyCg7HpW/
T9naeOERksa+Rj4R57IF1l4e5oiiGJo9QmaKZcsCsXrREJCycrlEtMqXfSPy+bi5
0yDZE/Qm1dwu13+OXOsRvkoNYjO8Mzo9K8wU12hMqN0a2bu6a7QjRWxnYW1hbCAy
MDQ4IDx0ZXN0MjA0OEBleGFtcGxlLm9yZz6IXgQTEQIAHgUCQsgiCgIbAwYLCQgH
AwIDFQIDAxYCAQIeAQIXgAAKCRBI6c1W/qZo29PDAKCG724enIxRog1j+aeCp/uq
or6mbwCePuKy2/1kD1FvnhkZ/R5fpm+pdm25Ag0EQsgiIhAIAJI3Gb2Ehtz1taQ9
AhPY4Avad2BsqD3S5X/R11Cm0KBE/04D29dxn3f8QfxDsexYvNIZjoJPBqqZ7iMX
MhoWyw8ZF5Zs1mLIjFGVorePrm94N3MNPWM7x9M36bHUjx0vCZKFIhcGY1g+htE/
QweaJzNVeA5z4qZmik41FbQyQSyHa3bOkTZu++/U6ghP+iDp5UDBjMTkVyqITUVN
gC+MR+da/I60irBVhue7younh4ovF+CrVDQJC06HZl6CAJJyA81SmRfi+dmKbbjZ
LF6rhz0norPjISJvkIqvdtM4VPBKI5wpgwCzpEqjuiKrAVujRT68zvBvJ4aVqb11
k5QdJscAAwUH/jVJh0HbWAoiFTe+NvohfrA8vPcD0rtU3Y+siiqrabotnxJd2NuC
bxghJYGfNtnx0KDjFbCRKJVeTFok4UnuVYhXdH/c6i0/rCTNdeW2D6pmR4GfBozR
Pw/ARf+jONawGLyUj7uq13iquwMSE7VyNuF3ycL2OxXjgOWMjkH8c+zfHHpjaZ0R
QsetMq/iNBWraayKZnWUd+eQqNzE+NUo7w1jAu7oDpy+8a1eipxzK+O0HfU5LTiF
Z1Oe4Um0P2l3Xtx8nEgj4vSeoEkl2qunfGW00ZMMTCWabg0ZgxPzMfMeIcm6525A
Yn2qL+X/qBJTInAl7/hgPz2D1Yd7d5/RdWaISQQYEQIACQUCQsgiIgIbDAAKCRBI
6c1W/qZo25ZSAJ98WTrtl2HiX8ZqZq95v1+9cHtZPQCfZDoWQPybkNescLmXC7q5
1kNTmEU=
=8QM5
-----END PGP PUBLIC KEY BLOCK-----
') As pubkey) As keys;
```

٣. قد ترى بعد ذلك السجلات في الجدول:

```
SELECT username, cc FROM testuserscards;
```

٤. الآن، يمكنك استخدام pgp_keyidto التحقق من المفتاح العام الذي استخدمته لتشفير البيانات:

```
SELECT pgp_key_id(dearmor('
-----BEGIN PGP PUBLIC KEY BLOCK-----
Version: GnuPG v1.4.1 (GNU/Linux)
mQGiBELIIgoRBAC1onBpxKYgDvrgCaUWPY34947X3ogxGOfCN0p6Eqrx+2PUhm4n
vFvmczpMT4iDc0mUO+iwnwsEkXQI1eC99g8c0jnZAvzJZ5miAHL8hukMAMfDkYke
5aVvcPPc8uPDlItpszGmH0rM0V9TIt/i9QEXetpyNWhk4jj5qnohYhLeZwCgkOdO
RFAdNi4vfFPivvtAp2ffjU8D/R3x/UJCvkzi7i9rQHGo313xxmQu5BuqIjANBUij
8IE7LRPI/Qhg2hYy3sTJwImDi7VkS+fuvNVk0d6MTWplAXYU96bn12JaD21R9sKl
Fzcc+0iZI1wYA1PczisUkoTISE+dQFUsoGHfpDLhoBuesXQrhBavI8t8VPd+nkdt
J+oKA/9iRQ87FzxdYTkh2drrv69FZHc3Frsjw9nPcBq/voAvXH0MRilqyCg7HpW/
T9naeOERksa+Rj4R57IF1l4e5oiiGJo9QmaKZcsCsXrREJCycrlEtMqXfSPy+bi5
0yDZE/Qm1dwu13+OXOsRvkoNYjO8Mzo9K8wU12hMqN0a2bu6a7QjRWxnYW1hbCAy
MDQ4IDx0ZXN0MjA0OEBleGFtcGxlLm9yZz6IXgQTEQIAHgUCQsgiCgIbAwYLCQgH
AwIDFQIDAxYCAQIeAQIXgAAKCRBI6c1W/qZo29PDAKCG724enIxRog1j+aeCp/uq
or6mbwCePuKy2/1kD1FvnhkZ/R5fpm+pdm25Ag0EQsgiIhAIAJI3Gb2Ehtz1taQ9
AhPY4Avad2BsqD3S5X/R11Cm0KBE/04D29dxn3f8QfxDsexYvNIZjoJPBqqZ7iMX
MhoWyw8ZF5Zs1mLIjFGVorePrm94N3MNPWM7x9M36bHUjx0vCZKFIhcGY1g+htE/
QweaJzNVeA5z4qZmik41FbQyQSyHa3bOkTZu++/U6ghP+iDp5UDBjMTkVyqITUVN
gC+MR+da/I60irBVhue7younh4ovF+CrVDQJC06HZl6CAJJyA81SmRfi+dmKbbjZ
LF6rhz0norPjISJvkIqvdtM4VPBKI5wpgwCzpEqjuiKrAVujRT68zvBvJ4aVqb11
k5QdJscAAwUH/jVJh0HbWAoiFTe+NvohfrA8vPcD0rtU3Y+siiqrabotnxJd2NuC
bxghJYGfNtnx0KDjFbCRKJVeTFok4UnuVYhXdH/c6i0/rCTNdeW2D6pmR4GfBozR
Pw/ARf+jONawGLyUj7uq13iquwMSE7VyNuF3ycL2OxXjgOWMjkH8c+zfHHpjaZ0R
QsetMq/iNBWraayKZnWUd+eQqNzE+NUo7w1jAu7oDpy+8a1eipxzK+O0HfU5LTiF
Z1Oe4Um0P2l3Xtx8nEgj4vSeoEkl2qunfGW00ZMMTCWabg0ZgxPzMfMeIcm6525A
Yn2qL+X/qBJTInAl7/hgPz2D1Yd7d5/RdWaISQQYEQIACQUCQsgiIgIbDAAKCRBI
6c1W/qZo25ZSAJ98WTrtl2HiX8ZqZq95v1+9cHtZPQCfZDoWQPybkNescLmXC7q5
1kNTmEU=
=8QM5
-----END PGP PUBLIC KEY BLOCK-----'));
```

يظهر ناتج طلب البحث هذا أن المفتاح العام التالي هو من قام بتشفير البيانات:

```
pgp_key_id
------------------
2C226E1FFE5CC7D4
(1 row)
```

الخطوة التالية هي التحقق مما إذا كان المفتاح العام الذي حصلت عليه قد استخدم في تشفير البيانات في الجدول:

```
hrdb=# SELECT username, pgp_key_id(cc) As keyweused FROM
testuserscards;
 username  |  keyweused
----------+------------------
 robby | 2C226E1FFE5CC7D4
 artoo | 2C226E1FFE5CC7D4
```

وأخيرا، فك تشفير البيانات باستخدام المفتاح الخاص الذي يطابق المفتاح العام الذي استخدمته في تشفير البيانات:

```
SELECT username, pgp_pub_decrypt(cc, keys.privkey) As ccdecrypt
FROM testuserscards
 CROSS JOIN
 (SELECT dearmor('-----BEGIN PGP PRIVATE KEY BLOCK-----
Version: GnuPG v1.4.1 (GNU/Linux)
lQG7BELIIgoRBAC1onBpxKYgDvrgCaUWPY34947X3ogxGOfCN0p6Eqrx+2PUhm4n
vFvmczpMT4iDc0mUO+iwnwsEkXQI1eC99g8c0jnZAvzJZ5miAHL8hukMAMfDkYke
5aVvcPPc8uPDlItpszGmH0rM0V9TIt/i9QEXetpyNWhk4jj5qnohYhLeZwCgkOdO
RFAdNi4vfFPivvtAp2ffjU8D/R3x/UJCvkzi7i9rQHGo313xxmQu5BuqIjANBUij
8IE7LRPI/Qhg2hYy3sTJwImDi7VkS+fuvNVk0d6MTWplAXYU96bn12JaD21R9sKl
Fzcc+0iZI1wYA1PczisUkoTISE+dQFUsoGHfpDLhoBuesXQrhBavI8t8VPd+nkdt
J+oKA/9iRQ87FzxdYTkh2drrv69FZHc3Frsjw9nPcBq/voAvXH0MRilqyCg7HpW/
T9naeOERksa+Rj4R57IF1l4e5oiiGJo9QmaKZcsCsXrREJCycrlEtMqXfSPy+bi5
0yDZE/Qm1dwu13+OXOsRvkoNYjO8Mzo9K8wU12hMqN0a2bu6awAAn2F+iNBElfJS
8azqO/kEiIfpqu6/DQG0I0VsZ2FtYWwgMjA0OCA8dGVzdDIwNDhAZXhhbXBsZS5v
cmc+iF0EExECAB4FAkLIIgoCGwMGCwkIBwMCAxUCAwMWAgECHgECF4AACgkQSOnN
Vv6maNvTwwCYkpcJmpl3aHCQdGomz7dFohDgjgCgiThZt2xTEi6GhBB1vuhk+f55
n3+dAj0EQsgiIhAIAJI3Gb2Ehtz1taQ9AhPY4Avad2BsqD3S5X/R11Cm0KBE/04D
29dxn3f8QfxDsexYvNIZjoJPBqqZ7iMXMhoWyw8ZF5Zs1mLIjFGVorePrm94N3MN
PWM7x9M36bHUjx0vCZKFIhcGY1g+htE/QweaJzNVeA5z4qZmik41FbQyQSyHa3bO
kTZu++/U6ghP+iDp5UDBjMTkVyqITUVNgC+MR+da/I60irBVhue7younh4ovF+Cr
VDQJC06HZl6CAJJyA81SmRfi+dmKbbjZLF6rhz0norPjISJvkIqvdtM4VPBKI5wp
gwCzpEqjuiKrAVujRT68zvBvJ4aVqb11k5QdJscAAwUH/jVJh0HbWAoiFTe+Nvoh
frA8vPcD0rtU3Y+siiqrabotnxJd2NuCbxghJYGfNtnx0KDjFbCRKJVeTFok4Unu
VYhXdH/c6i0/rCTNdeW2D6pmR4GfBozRPw/ARf+jONawGLyUj7uq13iquwMSE7Vy
NuF3ycL2OxXjgOWMjkH8c+zfHHpjaZ0RQsetMq/iNBWraayKZnWUd+eQqNzE+NUo
7w1jAu7oDpy+8a1eipxzK+O0HfU5LTiFZ1Oe4Um0P2l3Xtx8nEgj4vSeoEkl2qun
fGW00ZMMTCWabg0ZgxPzMfMeIcm6525AYn2qL+X/qBJTInAl7/hgPz2D1Yd7d5/R
dWYAAVQKFPXbRaxbdArwRVXMzSD3qj/+VwwhwEDt8zmBGnlBfwVdkjQQrDUMmV1S
```

```
EwyISQQYEQIACQUCQsgiIgIbDAAKCRBI6c1W/qZo25ZSAJ4sgUfHTVsG/x3p3fcM
3b5R86qKEACggYKSwPWCs0YVRHOWqZY0pnHtLH8=
=3Dgk
-----END PGP PRIVATE KEY BLOCK-----') As privkey) As keys;
username |  ccdecrypt
----------+------------------
robby    |  4111111111111111
artoo    |  4111111111111112
(2 rows)
```

هناك المزيد ...

بدلا من تحديد كل من المفاتحين من المفاتحين الخاص / العام بشكل صريح، يمكنك أيضا استخدام أداة تسمى "GPG" لتوليد المفاتيح العامة والخاصة وتصديرها واستخدامها في بوستجريسكل.

"GPG" متاح على كل من لينكس ومنصات ويندوز.

يمكنك استخدام الخطوات المتسلسلة التالية لتوليد "GPG" وتصديرها:

١- أولا، توليد مفاتيح:

```
gpg --gen-key
```

٢- بعد ذلك، راجع قائمة المفاتيح التي أنشأتها:

```
gpg --list-secret-keys

sec   1024D/D9ABCD1E 2014-06-17
uid                 aaaac
ssb    1024g/E0B81D3A 2014-06-17
```

٣- وأخيرا، تصدير كل من المفاتيح الخاصة والعامة:

```
gpg -a --export E0B81D3A > public.key
gpg -a --export-secret-keys D9ABCD1E > secret.key
```

كسر كلمات سر بوستجريسكل

العديد من قواعد البيانات بما في ذلك قواعد البيانات مفتوحة المصدر وكذلك تلك ذات الملكية و الحقوق، تأتي مع حسابات المستخدمين الافتراضية، ولهذه المخططات أيضا كلمات سر افتراضية. كلمات المرور هذه معروفة لدى الكثير في زمننا هذا، ومن المهم أن يحتفظ مسؤول قاعدة البيانات بكلمات مرور غير مستعارة لحسابات المستخدمين هذه. ومع ذلك، عادة ما يفضل المشرفون الاحتفاظ بكلمات مرور بسيطة أو السماح أحيانا بحفظ كلمات المرور الافتراضية. وهذا أمر يجب تجنبه في البيئة الإنتاجية لأن الإضرار هنا سيؤدي إلى ثغرة أمنية كبيرة وهذا شيء يمكن استغلاله من قبل القراصنة. ولهذا السبب، بدأت المنظمات في تنفيذ سياسة قوية بشأن كلمة المرور.

القاعدة الشائعة في سياسات كلمة المرور هي تجميعة بين مجموعة من الأحرف الأبجدية الرقمية وعدد قليل من الأحرف الخاصة لفرض كلمة مرور قوية. من المهم الاهتمام بطول كلمة مرور لا يقل عن ثمانية أحرف. في المثال التالي، سنرى كيف يمكن كسر كلمات المرور الضعيفة ومدى أهمية فرض سياسة كلمة مرور قوية.

كيف ينجز ذلك...

هنا، نحن نقدم سيناريو حيث أننا سوف نظهر كيف يمكن أن تكسر كلمات المرور الضعيفة بسهولة. ولأغراض العرض التوضيحي، سننشئ مستخدمين اثنين: مستخدم واحد لديه كلمة مرور مكونة من أرقام فقط، ومستخدم آخر لديه كلمة مرور مكونة من أحرف أبجدية فقط. نفذ الخطوات التالية لإنشاء المستخدمين:

١. أولا، أنشئ مستخدما وحدد كلمة مرور له:

```
create user xyz with password '123';
create user john with password 'good';
```

بعد ذلك، احصل على كلمات المرور المشفرة للمستخدمين من جدول pg_shadowcatalog:

```
select usename as useraccount,passwd as "password"
from pg_shadow
where length(passwd)>1 order by usename;

 useraccount     |                password
-----------------+-------------------------------------
 john            | md5bec8abeddc3e1513db64c184a6bf85c9
 xyz             | md5adf47922f0bdb6b9a520ed2d43622d14
(2 rows)
```

٢. ستكون الخطوة التالية هي استخدام أداة MD5 مفكك شفرات كلمة المرور. يمكنك استخدام أدوات مثل " Cain" و"MD Crack"، "Abel"، وهلم جرا. ومع ذلك، في حالتنا، سنقوم باستخدام موقع على شبكة الانترنت يسمى www.md5online.org، وسوف نستخدم مرفق MD5 فكك الشفرات على الانترنت، كما هو موضح هنا:

MD5 Decrypter

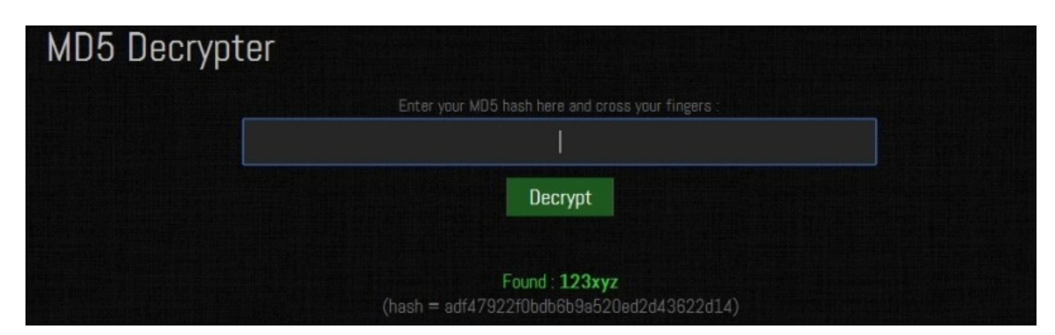
Enter your MD5 hash here and cross your fingers :

Decrypt

Found : **goodjohn**

(hash = bec8abeddc3e1513db64c184a6bf85c9)

في الصورة أعلاه، أدخلنا تجزئة MD5 للمستخدم john وإيجاد كلمة المرور الخاصة به. الاتفاقية التي نراها هنا هي أن يتم عرض كلمة المرور أولا، يليها اسم المستخدم. على سبيل المثال، نحصل على **goodjohn** حيث كلمة السر هو good للمستخدم john.



وبالمثل، في الصورة السابقة، أدخلنا تجزئة md5 للمستخدم "xyz" وإيجاد كلمة المرور الخاصة به. الاتفاقية التي نراها هنا أيضا، تماما كما في الحالة السابقة، حيث تعرض كلمة المرور أولا، يليها اسم المستخدم. على سبيل المثال، نحصل على 123xyz حيث أن كلمة المرور هي "١٢٣" للمستخدم "xyz".

كيف تعمل...

في السيناريو السابق، يمكنك أن ترى أن أي كلمة مرور أقل من ستة أحرف في الطول تكون عرضة للكسر بسرعة. لذلك، من المهم فرض سياسة كلمة مرور قوية وتثقيف المستخدمين بقيمة فعالية كلمة المرور القوية.

يلعب طول كلمة المرور دورا رئيسيا في فرض كلمة مرور قوية. ويعد طول كلمة المرور، والأكثر من ذلك هو الوقت الذي يتطلبه العمل لكسرها. ويستند تكسير كلمات المرور على طريقتين:

- **القوة الغاشمة** "Brute force": تتطلب هذه الطريقة أن تحاول كل طريقة ممكنة يجب عليك القيام بها من أجل كسر كلمة مرور. هناك المعروفة بمفرقعات كلمة المرور، مثل "Cain" و"" Abel"، "LophtCrack, "Hydra"، وهلم جرا، التي يمكنها استخدام طريقة القوة الغاشمة لمعرفة كلمة مرور. هذا الأسلوب هو مناسب لاختبار كلمات المرور القصيرة فقط.
- **هجوم القاموس** "Dictionary attack": هذا الأسلوب ينطوي على استخدام قائمة القاموس من أجل كسر كلمة المرور. هنا، يتم أخذ كل كلمة من القاموس بالتسلسل، وتحويلها إلى تجزئة، ومن ثم مطابقتها مع تجزئة النظام. إذا كانت متطابقة، فقد كسرت كلمة المرور، وإلا سينتقل إلى الكلمة التالية في القاموس وهلم جرا.

٣- النسخ الاحتياطي والاسترداد

في هذا الفصل، سنغطي الوصفات التالية:

- نسخة احتياطية منطقية لقاعدة بيانات بوستجريسكل واحدة
- النسخ الاحتياطي المنطقي لجميع قواعد البيانات بوستجريسكل
- نسخة احتياطية منطقية من كائنات محددة
- نظام النسخ الاحتياطي مستوى النظام
- أخذ نسخة احتياطية للقاعدة
- النسخ الاحتياطي المادي الساخن والأرشيف المستمر
- استرداد نقطة محددة في الوقت
- استعادة قواعد البيانات وكائنات قاعدة بيانات محددة

مقدمة

النسخ الاحتياطي والاستعادة عادة ما يشير إلى حماية قاعدة البيانات ضد فقدان البيانات وتمكن من استعادة البيانات في حالة فقدان البيانات. النسخ الاحتياطي، بعبارات بسيطة، هو نسخة من بيانات قاعدة البيانات الخاصة بك.

وتنقسم النسخ الاحتياطية إلى عنصرين:

- **النسخ الاحتياطي المنطقي**: يشير النسخ الاحتياطي المنطقي إلى ملف التفريغ الذي تم إنشاؤه بواسطة pg_dumputility والتي يمكن استخدامها لاستعادة قاعدة البيانات في حالة فقدان البيانات أو الحذف العرضي لكائن قاعدة بيانات مثل جدول. و pg_dumputility هو أداة محددة بوستجريسكل التي يمكن تشغيلها على سطر الأوامر، الأمر الذي ينشئ اتصال بقاعدة البيانات ويبدأ النسخ الاحتياطي المنطقي.
- **النسخ الاحتياطية الفزيائي**: النسخ الاحتياطي الملموس يشير إلى النسخ الاحتياطي على مستوى نظام التشغيل لقاعدة البيانات والدليل والملفات المرتبطة به.

ومن الضروري وضع استراتيجية تخطيط لتنفيذ النسخ الاحتياطية. وهذا أمر مرغوب فيه من وجهة نظر سيناريو الاستعادة، وفي حالة حدوث مثل هذه الحالة، فإن نوع النسخ الاحتياطية التي نبدأها سيؤثر على نوع الاستعادة الممكن.

نسخة احتياطية منطقية لقاعدة بيانات بوستجريسكل واحدة

تستخدم الأداة المساعدة pg_dump للاحتفاظ بقاعدة بيانات بوستجريسكل. وهي تجعل النسخ الاحتياطية متسقة حتى لو كانت قاعدة البيانات المستخدمة من قبل معاملات أخرى. يمكن إنشاء "Dumps" في السكريب أو في تنسيقات ملفات الأرشيف. مقالب السكريبت عادة ما تكون ملفات نصية عادية تحتوي على أوامر "SQL" المطلوبة لإعادة بناء قاعدة البيانات إلى الحالة التي كانت فيها في الوقت الذي حفظت فيه. ويمكن أيضا أن تستخدم مقالب السكربت لإعادة بناء قاعدة البيانات على الآلات ومعماريات الأخرى.

الاستعداد للعمل

يرجى ملاحظة أن كلمة تفريغ تستخدم هنا كمرادف للنسخ الاحتياطي.

وتعتبر الأداة المساعدة pg_dump نسخة احتياطية منطقية لأنها تصنع نسخة من البيانات في قاعدة البيانات عن طريق إفراغ محتويات كل جدول.

السطر التالي يعرض التركيب الأساسي لعمل نسخة احتياطية منطقية لقاعدة بيانات واحدة:

```
pg_dump -U username -W -F t database_name > [Backup Location Path]
```

وهنا شرح استخدام الخيارات المستخدمة مع الأمر pg_dump :

- **محدد U:** يحدد U- مستخدم قاعدة البيانات الذي ينشئ الاتصال. حيث أن pg_dump هي أداة سطر الأوامر، ونحن بحاجة إلى تحديد اسم المستخدم الذي يمكّن الأداة المساعدة pg_dump من ضبط اتصال بقاعدة بيانات
- **محدد W:** هذا الخيار ليس إلزاميا. يؤدي هذا الخيار إلى فرض pg_dump أن تطالب بكلمة المرور قبل الاتصال بخادم قاعدة بيانات بوستجريسكل. بعد الضغط على إنتر، سيقوم pg_dump بطلب كلمة مرور مستخدم قاعدة البيانات ليتم بدء الاتصال.
- **محدد F:** يحدد F- تنسيق ملف الإخراج الذي سيستخدم. حددنا خيار t مع F- لأن ملف الإخراج الناتج سيكون ملف أرشيف بصيغة tar.

هناك الكثير من الخيارات الأخرى المتاحة مع الأمر pg_dump؛ ومع ذلك، ولغرضنا هنا، فنحن سنستخدم الخيارات السابقة.

كيف ينجز ذلك...

هنا، في حالتنا، لدينا قاعدة بيانات اسمها dvdrenta والتي نحن بحاجة إلى توليد لها نسخة احتياطية منطقية.

هناك طريقتان حيث يمكن بدء التفريغ المنطقي في بوستجريسكل:

- النهج الأول هو استخدام الأداة المساعدة سطر الأوامر pg_dump لعمل تفريغ منطقي لقاعدة بيانات. هنا، نستخدم الأداة المساعدة pg_dump لعمل نسخة احتياطية من قاعدة بيانات dvdrenta في ملف الإخراج المسمى dvdrental.tar، الذي سيحفظ في الدليل الفرعي abcd من الدليل الرئيسي home:

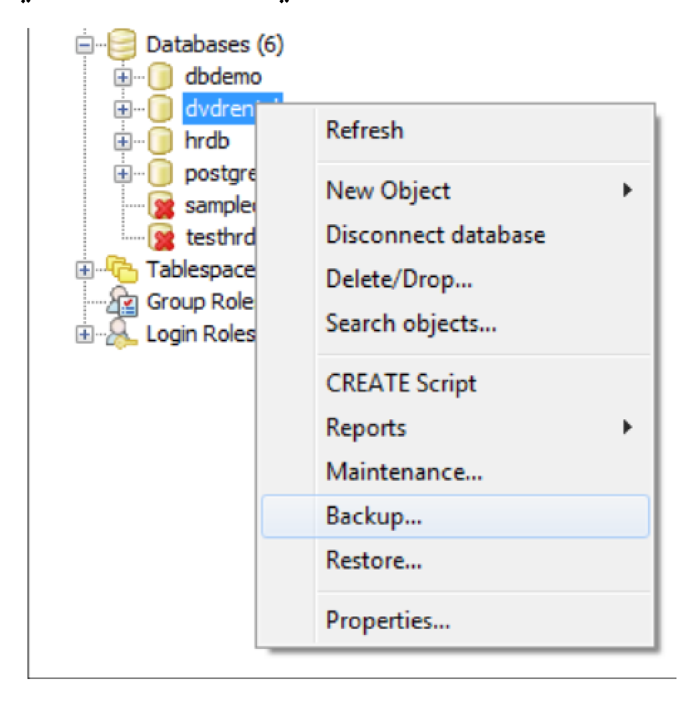
```
pg_dump -U postgres -W -F t dvdrental >
/home/abcd/dvdrental.tar
```

- الخيار الثاني هو استخدام أداة pgAdmin GUI لإجراء نسخ احتياطي لقاعدة بيانات فردية. هنا، سوف سنشرح لك كيفية النسخ الاحتياطي قاعدة بيانات dvdrenta باستخدام أداة pgAdmin:

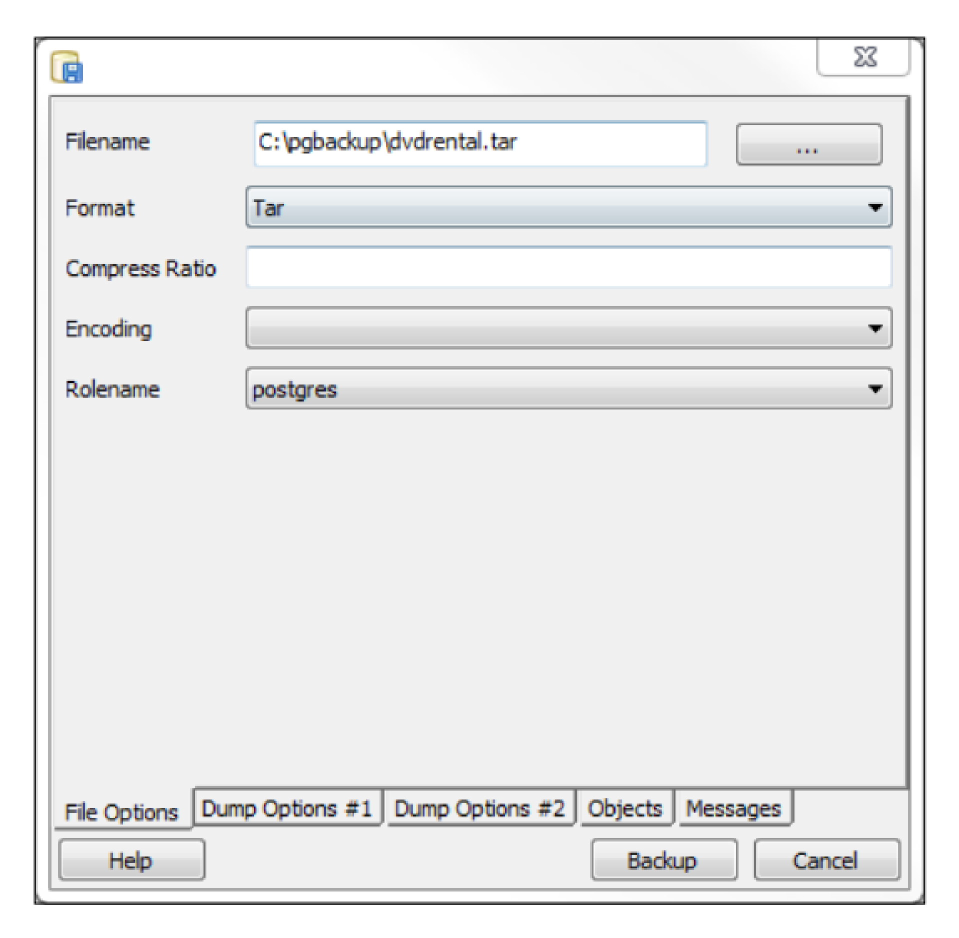
١. أولا، قم بتشغيل أداة pgAdmin GUI.

٢. انقر على قائمة Databases تحت الجزء "متصفح الكائن"في الجانب الأيمن من النافذة، حدد قاعدة بيانات dvdrenta ، وانقر بالزر الايمن على ذلك.

٣. ثم حدد خيار النسخ الاحتياطي ...، كما هو موضح في الصورة التالية:

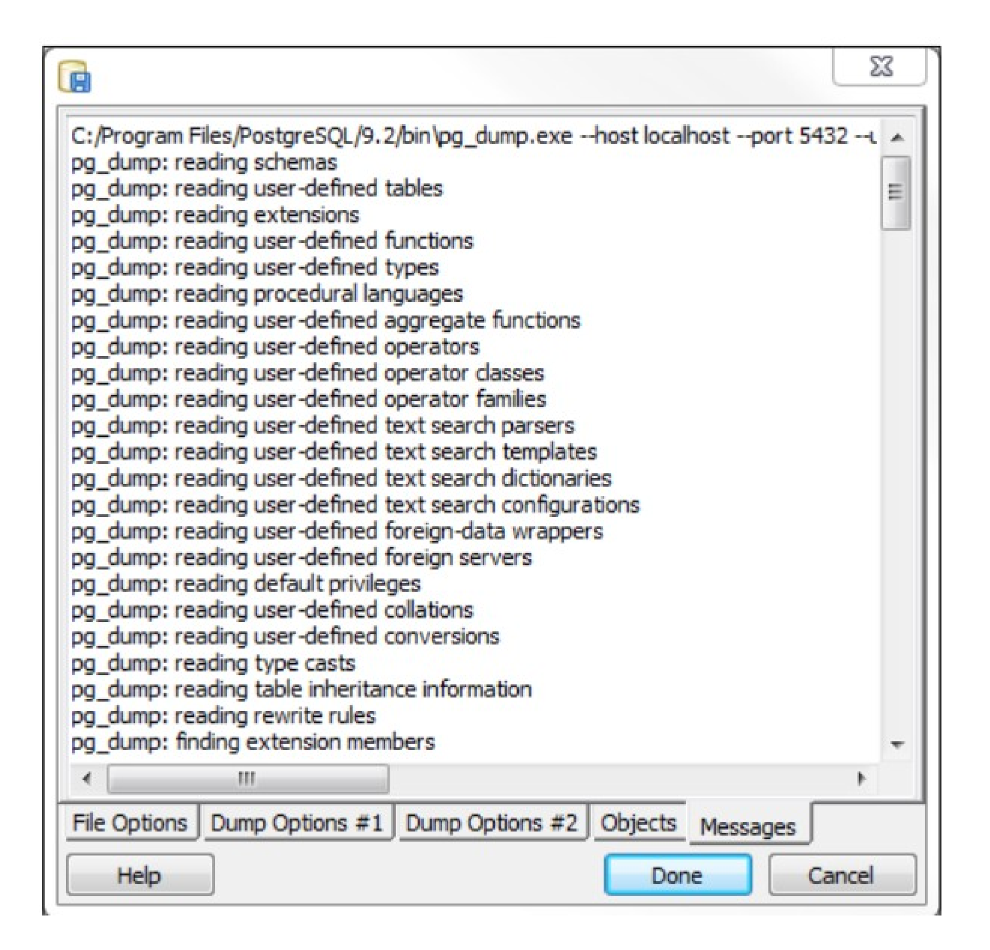


٤. عندما يحدد خيار النسخ الاحتياطي، سيظهر مربع الحوار، كما هو موضح في الصورة التالية، وسوف يكون عليك إدخال اسم للتفريغ المنطقي والذي سيكون ضروريا لاستعادة قاعدة بيانات أو جدولا معينا في حالة الفشل. هنا، قمنا بتسميته dvdrental.tar وسيخزن في الدليل pgbackup على القرص C.



٥. انقر على زر Backup لتوليد تفريغ قاعدة بيانات dvdrenta منطقية.

٦. حال النقر على زر Backup ، ستبدأ عملية إنشاء تفريغ منطقي ، وعرض الرسائل ذات الصلة بالعملية والتي يمكن مشاهدتها في الصورة التالية:



كيف تعمل...

يعمل أمر pg_dump بواسطة تنفيذ تعليمات SQL اتجاه قاعدة بيانات لتفريغ البيانات. أثناء تشغيل الأمر pg_dump، فإنه يقوم بقفل الجداول التي يقوم بتفريغها. وذلك من أجل التأكد من منع عمليات DDL أن تعمل في الجداول أثناء عملية التفريغ من أجل ضمان اتساق البيانات.

تكون التفريغات التي تنشئ بواسطة pg_dump متسقة داخليا؛ وهذا هو، حيث يمثل تفريغ لقطة من قاعدة البيانات في الوقت الذي بدأ فيه pg_dump بالعمل. إن الأداة المساعدة pg_dump لا تمنع أي عمليات أخرى لقاعدة البيانات أثناء تنفيذها. وفي هذه الحالة، الاستثناءات الوحيدة هي تلك العمليات التي تتطلب قفلا حصريا للتشغيل.

يمكن لأي مستخدم نظام أن يشغل pg_dump بشكل افتراضي، ولكن يجب أن يكون لدى المستخدم الذي يقوم بالاتصال ب بوستجريسكل حقوق "SELECT التحديد" لكل كائن قاعدة بيانات يراد إفراغه.

بما أن pg_dump يوفر أيضا خيارات اتصال قياسية لتحديد اتصال المضيف، فإنه يمكن استخدامه أيضا لتنفيذ النسخ الاحتياطية عن بعد من أي مضيف يسمح بإجراء اتصال بعيد كما هو محدد في ملف pg_hba.conf:

```
pg_dump -u postgres -h 192.168.16.54 -F c -f
dvdrental.sql.tar.gz dvdrental
```

في السيناريو السابق، نقوم بالاتصال بقاعدة بيانات dvdrental الموجودة في المضيف ذي عنوان ١٩٢.١٦٨.١٦.٥٤ وبدء نسخة احتياطية عن بعد لقاعدة بيانات dvdrental. وينشئ ملف تفريغ dvdrental.sql.tar.gz في الموقع الحالي حيث ينفذ الأمر pg_dump.

يمكنك الرجوع إلى الروابط التالية للحصول على معلومات أكثر تفصيلا حول الأداة المساعدة pg_dump:

- http://www.postgresql.org/docs/9.3/static/app-pgdump.html
- http://www.commandprompt.com/blogs/joshua_drake/2010/07/a_better_backup_with_postgre/sql_using_pg_dump
- http://www.postgresonline.com/special_feature.php?sf_name=postgresql83_pg_dumprestore_cheatsheet

النسخ الاحتياطي المنطقي لجميع قواعد البيانات بوستجريسكل

للنسخ الاحتياطي لكافة قواعد البيانات، يمكنك تشغيل الأمر الفردي pg_dump لكل قاعدة بيانات بالتتابع أو بالتوازي إذا كنت ترغب في تسريع عملية النسخ الاحتياطي:

- أولا، من عميل psql ، استخدم الأمر \l لإدراج كافة قواعد البيانات المتوفرة في الكتلة الخاصة بك.
- ثانيا، نسخ احتياطي لكل قاعدة بيانات فردية باستخدام الأمر pg_dump، كما هو موضح في الوصفة السابقة.

النهج الآخر هو استخدام أداة pg_dumpall لعمل نسخة احتياطية من كافة قواعد البيانات في دفعة واحدة.

كيف ينجز ذلك...

يمكنك استخدام الأمر pg_dump النسخ الاحتياطي لكل قاعدة بيانات في الخادم؛ ومع ذلك، pg_dump لا تفرغ المعلومات حول وصف الدور ومساحات الجداول. لتفريغ المعلومات الإجمالية، استخدم الأمر التالي:

```
pg_dumpall –g
```

لعمل نسخة احتياطية من كافة قواعد البيانات دفعة واحدة، يمكنك استخدام الأداة المساعدة pg_dumpall، كما يلي، في ويندوز:

```
pg_dumpall -U postgres > c:\pgbackup\all.sql
```

وبالمثل، لإجراء نسخ احتياطي لجميع قواعد البيانات دفعة واحدة في لينكس، استخدم الأمر pg_dumpall، كما يلي:

```
pg_dumpall -U postgres > /home/pgbackup/all.sql
```

كيف تعمل...

يقوم الأمر pg_dumpall بتصدير كافة قواعد البيانات، واحدا تلو الآخر، إلى ملف نصي واحد يمنعك من إجراء استعادة متوازية. إذا كنت تقوم بنسخ احتياطي لجميع قواعد البيانات بهذه الطريقة، فإن عملية الاستعادة ستستغرق الكثير من الوقت.

وقت معالجة تفريغ جميع قواعد البيانات يستغرق وقتا أطول من الوقت اللازم لتفريغ كل قاعدة بيانات على حدة، ولذلك نحن لا نعرف أي من قواعد البيانات التي تفرغ حاليا في لحظة معينة من الوقت.

لهذا السبب، يجب استخدام الأمر pg_dump لتفريغ كل قاعدة بيانات بشكل فردي ثم استخدم محدد g- من الأمر pg_dumpall للاحتفاظ بنسخة احتياطية من كافة بيانات المستخدم والمجموعة.

الأمر pg_dumpall يتطلب عموما أن يكون المستخدم الذي يقوم بتنفيذ السكريبت مستخدم جذر لبوستجريسكل. وذلك لأن الأمر pg_dumpall يتطلب النفاذ إلى كتالوجات نظام بوستجريسكل، كما أنه يفرغ الكائنات العامة وكذلك كائنات قاعدة البيانات.

هناك المزيد...

في بعض الأحيان، تريد إجراء نسخ احتياطي فقط لتعريفات كائن قاعدة البيانات، بحيث يمكنك استعادة المخطط فقط. هذا مفيد لمقارنة ما يتم تخزينه في قاعدة البيانات ضد التعاريف في البيانات أو أداة النمذجة.

ومن المفيد أيضا التأكد من أنه يمكنك إعادة إنشاء الكائنات في المخطط الصحيح ومساحة الجداول وقاعدة البيانات مع الملكية والأذونات الصحيحة.

لعمل نسخ احتياطي لكافة تعريفات الكائنات في كل قواعد البيانات بما في ذلك الأدوار ومساحات الجداول وقواعد البيانات ومخططات الجداول والفهارس والمشتغلات والوظائف والقيود وجهات النظر والملكية والامتيازات، يمكنك استخدام الأمر التالي في ويندوز:

```
pg_dumpall --schema-only > c:\pgdump\definitiononly.sql
```

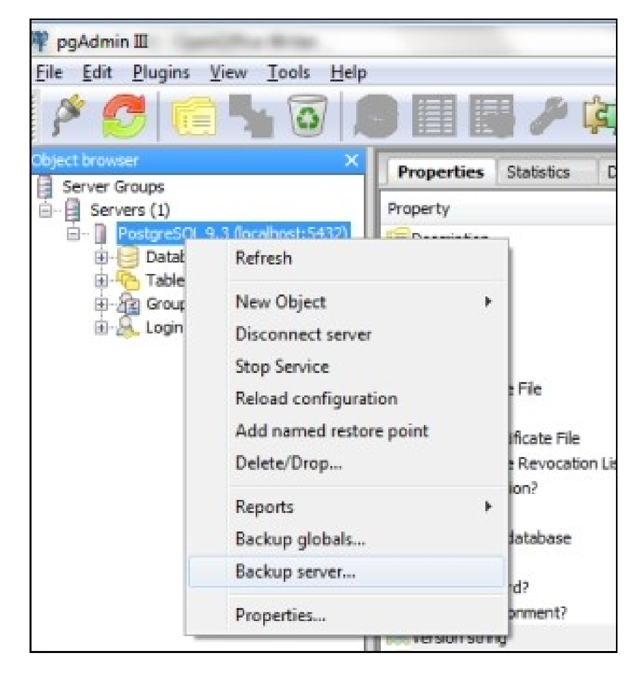
إذا كنت تريد الاحتفاظ بنسخة احتياطية من تعريف الدور فقط، استخدم الأمر التالي:

```
pg_dumpall --roles-only > c:\pgdump\myroles.sql
```

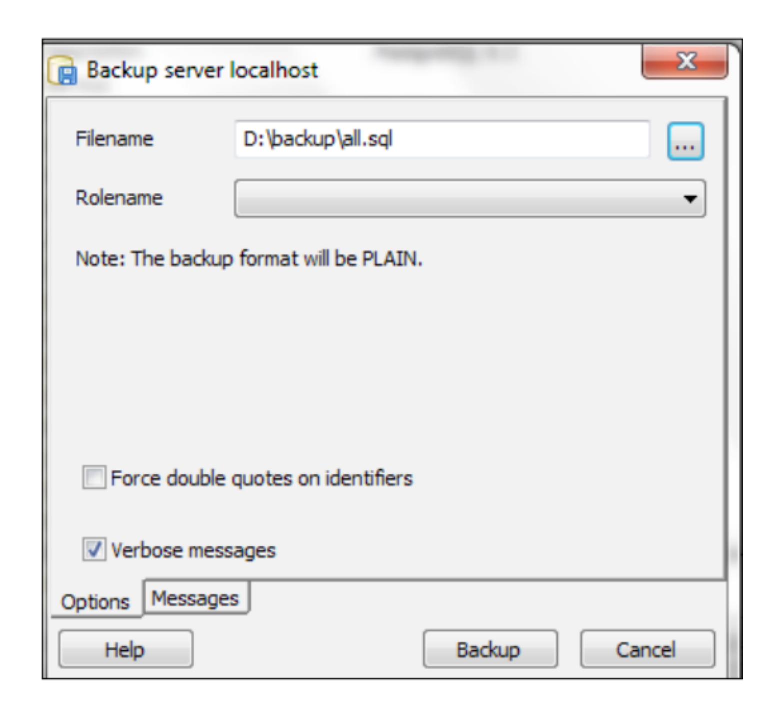
إذا كنت تريد إجراء نسخ احتياطي لتعريفات مساحة الجداول، استخدم الأمر التالي:

```
pg_dumpall --tablespaces-only > c:\pgdump\mytablespaces.sql
```

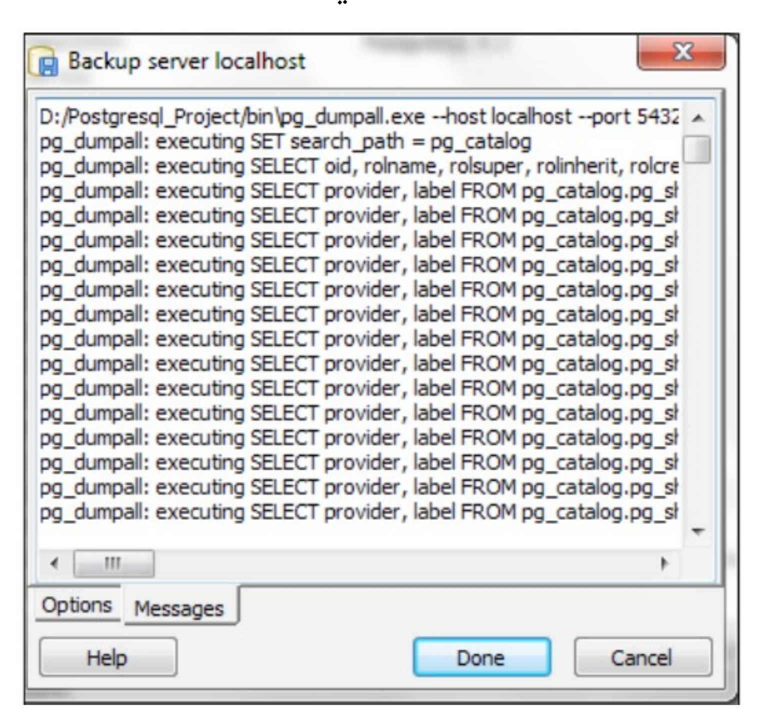
يمكنك أيضا استخدام أداة pgAdmin لتقوم بالنسخ الاحتياطي لكافة قواعد البيانات على الخادم، بما في ذلك الأدوار والمستخدمين والمجموعات، والجداول. يمكن تحديد هذا الخيار من خلال إطلاق أداة pgAdmin وتوسيع خيار قائمة الخوادم في الجزء الأيمن، والنقر بزر الماوس الأيمن على الخيار بوستجريسكل ٩.٣، ثم تحديد خيار خادم النسخ الاحتياطي ... كما هو موضح في لقطة الشاشة التالية:



بعد النقر فوق خيار النسخ الاحتياطي للخادم Backup server ... سيظهر مربع حوار، كما هو موضح في الصورة التالية، وسوف يطلب منك تسمية ملف تفريغ منطقي الذي سوف يحتوي على تعريف كافة الكائنات وقواعد البيانات التي سيصنع لها نسخة احتياطية . هنا، all.sql هو اسم ملف تفريغ منطقي التي يمكن استخدامه في وقت لاحق لاستعادة البيانات وتعريفات الكائن في كافة قواعد البيانات في حالة فقدان البيانات بسبب مشاكل الأجهزة أو القرص.



بمجرد النقر على زر النسخ الاحتياطي، كما هو مبين في الصورة السابقة، سوف تبدأ عملية النسخ الاحتياطي المنطقي لجميع قواعد البيانات. وهذا ما يعادل إصدار الأمر pg_dumpall، ويمكن رؤية تأثير ذلك في الصورة التالية، التي تفتح عند النقر على زر النسخ الاحتياطي:



نسخة احتياطية منطقية من كائنات محددة

في بعض الأحيان، هناك حالات حيث ترغب في نسخ احتياطي فقط لكائنات محددة من قاعدة البيانات، مثل الجداول.

توفر الأداة المساعدة pg_dump خيارات لنسخ احتياطي لكائنات قاعدة بيانات معينة، مثل الجداول.

كيف ينجز ذلك...

إذا كنت ترغب في الاحتفاظ بنسخة احتياطية من بعض الجداول المحددة في مخطط معين، يمكنك استخدام الأمر pg_dump كما يلي:

```
pg_dump -h localhost -p 5432 -U agovil -F c -b -v -f
"C:\pgbak\testdb_test.backup" -t case.test postgres
```

في الأمر السابق، قمنا بنسخ احتياطي لجدول يسمى test الذي يتواجد في مخطط الحالة في قاعدة بيانات بوستجريسكل.

كيف تعمل...

هناك العديد من الحالات التي تتطلب منك إجراء نسخ احتياطي لجداول معينة فقط. بعض هذه الحالات المذكورة هنا:

- إن كنت ترغب في عمل نسخة احتياطية لكافة الجداول التي هي جزء من مساحة جداول معينة. في هذه الحالة، من الممكن أن تحتوي مساحة الجداول على كائنات لأكثر من قاعدة بيانات واحدة؛ وبالتالي، عليك أن تحدد قواعد البيانات التي تحتاج إلى إلقاء تلك الجداول.

يتيح لك الإجراء التالي تفريغ كافة الجداول الموجودة على مساحة جدول واحد وقاعدة بيانات واحدة فقط:

١. قم بإنشاء ملف باسم table_tablespace_dump.sql، والذي يحتوي على الأمر SQL التالي الذي يقوم باستخراج قائمة الجداول في مساحة جدول.

```
SELECT 'pg_dump' UNION ALL
SELECT '-t ' || spcname || '.' || relname FROM pg_class t
JOIN pg_tablespace ts
ON reltablespace = ts.oid AND spcname = :TSNAME
JOIN pg_namespace n ON n.oid = t.relnamespace
WHERE relkind = 'r'
UNION ALL
SELECT '-F c > dumpfile'; -- dumpfile is the name of the output file
```

٢. تنفيذ ما يلي لبناء السكريبت pg_dump:

```
psql -t -v TSNAME="'my_tablespace'" -f
table_tablespace_dump.sql > myts
```

٣. من خادم قاعدة البيانات، قم بتفريغ الجداول في مساحة الجدول، بما في ذلك البيانات والتعاريف:

```
chmod 700 myts
./myts
```

- هناك حالة أخرى حيث توجد مخططات متعددة لديها جداول مهمة ذات تسميات متشابهة. و ترغب في نسخ احتياطي الجداول التي لها نفس الأسماء من مخططات مختلفة.

السيناريو التالي يمكنك من نسخ احتياطي للجداول ذات التسمية المتشابهة من مخططات مختلفة. يمكنك استخدام الاستعلام التالي الذي يقوم بإنشاء أمر pg_dump لإجراء نسخ احتياطي لأي جدول غير موجود على العام أو pg_catalog يحتوي على الكلمة المفتاحية الإيجار "rent " كلاحقة جزء من اسم الجدول:

```
SELECT 'pg_dump ' || ' -h localhost -p 5432 -U postgres -F c -b
-v -f "/pgbak/dvdrental_keytbls.backup" ' ||
array_to_string(ARRAY(SELECT '-t ' || table_schema || '.' ||
table_name FROM information_schema.tables
WHERE table_name LIKE '%_rent' AND table_schema NOT IN
('pg_catalog','public' ) ), ' ') || 'dvdrental';
```

- حالة أخرى عندما تريد النسخ الاحتياطي للجداول التي تغيرت مؤخرا في قاعدة البيانات.

نحن بحاجة إلى استخدام إحصاءات التنظيف vacuum statistics لهذا الغرض. تعتمد إحصاءات التنظيف على افتراض أن عملية التنظيف سوف تحاول أن تمر على كل قاعدة البيانات والجداول وتنظف كل البيانات المتغيرة منذ تشغيل آخر عملية تنظيف سابقة. هذه الآلية سوف تعمل فقط إذا كانت أوتوفاكوم autovacuum مفعلة. سيقوم الاستعلام التالي بإنشاء أمر pg_dump يقوم بالنسخ الاحتياطي لكافة الجداول في قاعدة بيانات dvdrental التي تم تحليلها تلقائيا في اليوم الماضي:

```
SELECT 'pg_dump ' || ' -h localhost -p 5432 -U postgres -F c -b
-v -f "C:/pgbak/dvdrental_changed_keytbls.backup" ' ||
array_to_string(ARRAY(SELECT '-t ' || schemaname || '.' ||
relname
FROM pg_ stat_user_tables
WHERE last_autoanalyze > ( CURRENT_TIMESTAMP - (INTERVAL '1
day') ) ) , ' ') || 'dvdrental';
```

النسخ الاحتياطي على مستوى ملفات النظام

خيار آخر للنسخ الاحتياطي وهو الاستفادة من أوامر نظام التشغيل وعمل نسخة احتياطية من الملفات التي يستخدمها بوستجريسكل لتخزين البيانات في قاعدة البيانات.

كيف ينجز ذلك...

أسهل طريقة للقيام بذلك هي إنشاء أرشيف لدليل بيانات بوستجريسكل أو الدليل الذي يحدده متغير البيئة PGDATA$، على النحو التالي:

```
tar -cvf backup.tar /home/abcd/psql/data
```

هنا، أنشأنا ملف أرشيف باسم backup.tar يحتوي على نسخة احتياطية من دليل البيانات.

كيف تعمل...

الفائدة الأساسية من إجراء النسخ الاحتياطي على مستوى نظام الملفات هو أن هذا الإجراء بسيط ومباشر. تحتاج ببساطة إلى نسخ دليل البيانات بأي أداة من الأدوات المساعدة للنسخ الاحتياطي في أنظمة يونكس ، مثل tar، التي تصنع ملف أرشيف الذي يمكن استخدامه مرة أخرى لإصلاح إذا تعطلت قاعدة البيانات.

ومع ذلك، هناك بعض القيود التي تستخدم الطريقة السابقة لإنشاء أرشيف لدليل البيانات:

- يجب إغلاق قاعدة البيانات تماما من أجل الحصول على نسخة احتياطية مفيدة. النسخ الاحتياطي لنظام الملفات له معنى فقط عندما تكون قاعدة البيانات في حالة متناسقة. لهذا السبب، تحتاج إلى إيقاف تشغيل قاعدة البيانات، ونتيجة لذلك، فإن كافة ملفات البيانات ستكون متزامنة وفي حالة متناسقة وهذا هو الوقت المناسب بأن تقوم بالنسخ الاحتياطي لنظام الملفات.
- مع النسخ الاحتياطي لنظام الملفات، لا يمكن إجراء نسخ احتياطي لقواعد بيانات معينة أو جداول فردية. يجب أن يكون يتم النسخ الاحتياطي لدليل البيانات بأكمله لاستعادة كاملة لنظام الملفات. ويرجع ذلك إلى أن العديد من الملفات المرتبطة بقاعدة بيانات معينة يصبح من الصعب ربط الملفات بقاعدة البيانات المعنية.

هناك المزيد...

هنا، سوف نتحدث عن كيفية اتخاذ النسخ الاحتياطية ل بوستجريسكل باستخدام لقطة LVM (اختصارلمدير حجم منطقي) (logical volume manager). وهذا ينطوي على أخذ لقطة مجمدة من وحدة التخزين التي تحتوي على قاعدة البيانات، ثم أخذ نسخة من دليل قاعدة البيانات من اللقطة إلى جهاز النسخ الاحتياطي، وأخيرا الإفراج عن لقطة المجمدة. سيعمل هذا حتى عند تشغيل خادم قاعدة البيانات. قبل البدء في التقاط اللقطة، يجب تنفيذ أمر CHECKPOINT في بوستجريسكل التي يمكنك من خلالها التأكد من أن النسخة الاحتياطية ستكون متسقة حتى لحظة " نقطة التحقق " "CHECKPOINT".

قبل أن نبدأ، نفترض أن لدينا نظام ملفات XFS منسق ونشط. ونفترض أيضا أن VG_POSTGRES / RV_DATA هو حجم البيانات الأساسي.

يجب على المستخدم الجذري تنفيذ الخطوات التالية من أجل إنشاء لقطة LVM واستخدامها:

١. إصدار أمر نقطة التحقق (CHECKPOINT) كمستخدم ذي صلاحيات مطلقة superuser على قاعدة بيانات بوستجريسكل، على النحو التالي:

```
postgres=# CHECKPOINT ;
```

٢. بقية الأوامر المذكورة هنا تحتاج إلى أن تنفذ من قبل المستخدم الجذر. في هذه الخطوة، سنقوم بإنشاء لقطة، على النحو التالي:

```
lvcreate -l 100%FREE -s -n snap VG_POSTGRES/RV_DATA
```

٣. قم بإنشاء الدليل الذي ترغب في تحميل اللقطة عليه، كما يلي:

```
mkdir /mnt/pg_snap
```

٤. ستكون الخطوة التالية هي تركيب اللقطة كملف XFS ، كما يلي:

```
mount -t xfs -o nouuid /dev/VG_POSTGRES/snap /mnt/pg_snap
```

٥. أدخل دليل اللقطة، كما هو موضح هنا:

```
cd /mnt/pg_snap/
```

٦. بعد ذلك، قم بعمل نسخة احتياطية من اللقطة باستخدام الأمر التالي:

```
tar —czvf /backup/ pgsql.$(date +"%m-%d%-%Y).tar.gz /mnt/pg_snap/
```

٧. بمجرد نقل لقطة النسخ الاحتياطي إلى خادم آخر، ستكون الخطوة التالية إلغاء تحميله وحذفه من الخادم الأصلي بصفة مستخدم جذر:

```
umount /mnt/pg_snap
lvremove VG_POSTGRES/snap
```

يتمثل النهج البديل في الشروع في استخدام متسق لصورة من دليل البيانات. وهذا ينطوي على أخذ لقطة مجمدة من وحدة التخزين التي تحتوي على قاعدة البيانات، ثم عمل نسخة من دليل قاعدة البيانات من لقطة إلى جهاز النسخ الاحتياطي، ثم ألإفراج عن اللقطة المجمدة. سيعمل هذا حتى عند تشغيل خادم قاعدة البيانات. قبل البدء في التقاط اللقطة، يجب تنفيذ أمر نقطة التحقق CHECKPOINT في بوستجريسكل التي يمكنك من خلالها التأكد من أن النسخة الاحتياطية ستكون متسقة حتى النفاذ إلى لحظة نقطة التحقق CHECKPOINT.

أخذ نسخة احتياطية قاعدية

أداة pg_basebackup تأخذ نسخ احتياطية قاعدية لقاعدة بيانات خادم بوستجريسكل.

يتم بدء هذه النسخ الاحتياطية دون التأثير على عملاء قاعدة بيانات بوستجريسكل الآخرين ويمكن استخدامها لكل من الاستعادة نقطة في الوقت، فضلا عن نقطة بداية لشحن السجل أو إلى تدفق خوادم النسخ.

كيف ينجز ذلك...

يمكنك استخدام الأمر pg_basebackup بالطريقة التالية:

```
$ pg_basebackup -h 192.168.10.14 -D /home/abcd/pgsql/data
```

هنا، نأخذ نسخة احتياطية قاعدية من الخادم الموجود في ١٩٢.١٦٨.١٠.١٤ وتخزينه في

```
/home/abcd/pgsql/datalocal directory.
```

كيف تعمل...

الأداة pg_basebackup تقوم بعمل نسخة ثنائية من ملفات نظام قاعدة البيانات مع ضمان أن يوضع النظام في وضع النسخ الاحتياطي تلقائيا. لاحظ أن النسخ الاحتياطية تتكون دائما من عنقود قاعدة البيانات بأكملها. لا يمكن عمل النسخ الاحتياطي لقواعد البيانات الفردي أو قواعد البيانات مع أداة pg_basebackup.

الأداة المساعدة pg_basebackup تبدأ النسخ الاحتياطي الذي يقوم بإجراء اتصال بقاعدة بيانات بوستجريسكل العادية ويستخدم بروتوكول النسخ لهذا الغرض. يجب أن يكون الشخص المنشئ للاتصال إما superuser أو مستخدم لديه امتياز REPLICATION. يجب أيضا تضبيط الخادم بما يكفي من MAX_wal_senders لتترك جلسة واحدة على الأقل متاحة للنسخ الاحتياطي.

النسخ الاحتياطي فيزيائية الساخن والأرشيف المستمر

في هذه الوصفة، سوف نتحدث عن أخذ نسخة احتياطية فيزيائية ساخنة مع الأرشفة المستمرة في المكان. النسخ الاحتياطي الفيزيائي الساخن هو النسخ الاحتياطي عبر الإنترنت online التي تؤخذ أثناء وجود معاملات نشطة بقاعدة البيانات. وعلى الرغم من أن لدينا النسخ الاحتياطي المادي على الإنترنت ذات الصلة التي من خلالها يمكننا استعادة قاعدة البيانات، فإنه يمكن استعادة البيانات فقط قبل وقت النسخ الاحتياطي. وسيتم تفويت أي معاملات لاحقة قد تكون قد سجلت في قاعدة البيانات بعد الانتهاء من النسخ الاحتياطي. ومن أجل أن تكون قادر على استعادة قاعدة البيانات حتى حالتها الحالية، سوف تحتاج إلى تطبيق المحفوظات التي تم إنشاؤها بعد الانتهاء من النسخ الاحتياطي. لهذا السبب، نحن بحاجة إلى تفعيل الأرشفة المستمرة.

كيف ينجز ذلك...

هناك سلسلة من الخطوات التي يجب القيام بها من أجل الحصول على النسخ الاحتياطي الفيزيائي الساخن والأرشيف المستمر في المكان:

١. وتتمثل الخطوة الأولى في تفعيل أرشفة سجل المتابعة المستمرة (التسجيل عبر الكتابة " write-ahead log". ويمكن القيام بذلك عن طريق إجراء التغييرات المعلمة التالية في ملف التضبيط postgresql.conf، الذي يتواجد في دليل البيانات، وهذا هو الدليل المحدد من قبل متغير البيئة $PGDATA:

```
wal_level = hot_standby
archive_mode = on
archive_command = 'test ! -f
/home/abcd/pgsql/backup_in_progress || (test ! -f
/home/abcd/pgsql/archive/%f && cp %p
```

```
/home/abcd/pgsql/archive/%f)'
```

٢. وبمجرد تنفيذ هذه التغييرات، تحتاج إلى تحديث خادم بوستجريسكل للتغييرات التي أدخلت على المعلمات المذكورة أعلاه لتصبح نافذة المفعول:

```
pg_ctl -d $PGDATA stop
pg_ctl -d $PGDATA start
```

٣. ستكون الخطوة التالية هي إنشاء دليل الأرشيف، كما يلي:

```
mkdir —p /home/abcd/pgsql/archive/
touch /home/abcd/pgsql/backup_in_progress
```

٤. بمجرد القيام بذلك، ستكون الخطوة التالية هي بدء عملية النسخ الاحتياطي باستخدام الأمر التالي:

```
psql -c "select pg_start_backup('hot_backup');"
```

٥. بعد ذلك، إجراء النسخ الاحتياطي نظام الملفات من دليل بيانات بوستجريسكل، على النحو التالي:

```
tar -cvf /home/abcd/pgsql/backup.tar /home/abcd/pgsql/data
```

٦. عند اكتمال النسخ الاحتياطي لنظام الملفات، ستكون الخطوة التالية للاتصال بقاعدة البيانات وإيقاف عملية النسخ الاحتياطي باستخدام الأمر التالي:

```
psql -c "select pg_stop_backup();"
```

٧. الآن وقد اكتملت عملية النسخ الاحتياطي، فإن الخطوة التالية ستكون الذهاب إلى موقع الأرشيف وتأكيد المحفوظات التي تم إنشاؤها. هذه المحفوظات، جنبا إلى جنب مع قاعدة النسخ الاحتياطي ، سوف تساعدك على استعادة البيانات إلى نقطة التدقيق الأخيرة أو أي نقطة في الزمن بعد حدوث النسخ الاحتياطي الأساسي وإلى النقطة التي لديك سجلات الأرشيف:

```
cd /home/abcd/pgsql/archive
[postgres@localhost archive]$ ls -ltrh
total 49M
-rw------- 1 postgres postgres 16M Jun 30 23:53
000000010000000000000009
-rw------- 1 postgres postgres 16M Jun 30 23:53
000000010000000000000008
-rw------- 1 postgres postgres 294 Jun 30 23:54
00000001000000000000000A.00000024.backup
```

```
-rw------- 1 postgres postgres 16M Jun 30 23:54
00000001000000000000000A
```

كيف تعمل...

النسخ الاحتياطي الفيزيائي أو النسخ الاحتياطي القاعدي يأخذ نسخة من كافة الملفات في قاعدة البيانات أو دليل البيانات؛ ومع ذلك، هذا وحدة لا يكفي كنسخ احتياطي وهناك خطوات أخرى تحتاج إلى أن تقوم بها أيضا. إن عملية النسخ الاحتياطي البسيطة لنظام الملفات الخاص بقاعدة البيانات، مع بقاء الخادم قيد التشغيل، ينتج نسخة غير متناسقة من الوقت من ملفات قاعدة البيانات. ومع ذلك، وفي السياق الحالي، تحتاج قواعد بيانات الإنتاج إلى أن تكون متاحة على مدار ٢٤/٧، وقادرة على القيام بالنسخ الاحتياطي، ولكن ليس من الممكن إيقاف قاعدة البيانات في كل مرة ومن ثم عمل النسخ الاحتياطي لنظام الملفات. إن هذه الاستراتيجية غير ممكنة. وعلاوة على ذلك، ينبغي لعملية النسخ الاحتياطي ضمان أن جميع التغييرات التي أجريت من وقت بدء النسخ الاحتياطي حتى الوقت الذي ينتهي فيه قد تعقبت وسجلت. هذه التغييرات تتعقب وتسجل في سجلات WAL، وبمجرد أرشفة التغييرات المسجلة في سجلات WAL، يمكن استخدام هذه السجلات لاحقا.

الآن وقد أصبح من حاجة الأعمال إلى اتخاذ النسخ الاحتياطية عبر الإنترنت لقواعد بيانات الإنتاج، تحتاج إلى التأكد من أن النسخ الاحتياطية التي تؤخذ على الإنترنت متسقة. لجعل النسخ الاحتياطية متسقة، تحتاج إلى إضافة إلى كل التغييرات التي حدثت من بداية إلى نهاية عملية النسخ الاحتياطي. هذا هو السبب وجود الخطوات ٤ و ٦ لوضع قوس حول خطوة النسخ الاحتياطي.

توضع التغييرات التي أجريت في دليل الأرشيف كمجموعة من سجلات المعاملات المؤرشفة ملفات log/WAL التسجيل عبر الكتابة. في الخطوة ٣، أنشأنا دليل الأرشيف. وتفعيل وضع الأرشيف، كما هو مذكور في الخطوة ١، يتطلب إعادة تشغيل قاعدة البيانات وتم ذلك في الخطوة ٢. في الخطوة ٣، يمكنك أن ترى أيضا أننا قد أنشأنا ملف باسم backup_in_progress. يؤدي وجود هذا الملف إلى تفعيل أو تعطيل عملية الأرشفة.

استرداد نقطة في الوقت أو الزمن

في كثير من الأحيان، سوف تواجه كمدير قاعدة بيانات حالات حيث قد تحتاج إلى استعادة قاعدة البيانات من نسخة احتياطية موجودة. قد يكون هذا بسبب متطلبات العمل أو قد يكون قد حذف جدول حرج، أو أن القرص الثابت الذي عليه قاعدة البيانات قد عطب وأصبح فاسدا. لأي سبب من الأسباب، قد تضطر إلى الذهاب إلى سيناريو استعادة قاعدة

البيانات. في هذه الوصفة، سنناقش الخطوات المطلوبة لاستعادة قاعدة البيانات في حالة الفشل وكيفية استخدام سجلات الأرشيف للقيام باستعادة نقطة في الوقت.

كيف ينجز ذلك...

هناك سلسلة من الخطوات التي تحتاج إلى تنفيذها إذا كنت بحاجة إلى استعادة قاعدة بيانات من النسخ الاحتياطي:

١. أولا، تحقق من حالة خادم قاعدة البيانات. إذا كان الخادم قيد التشغيل، قم بإيقافه باستخدام الأمر التالي:

```
pg_ctl -d $PGDATA stop
```

٢. بعد ذلك، انسخ دليل البيانات الموجودة وأي مساحات جداول موجودة إلى موقع مؤقت، إذا كان هناك أي شيء مطلوب من البنية الموجودة في وقت لاحق. في حالة وجود أزمة في المساحة، يجب عليك على الأقل النظر في الاحتفاظ بنسخة من محتوى الدليل الفرعي pg_xlog. وهذا أمر ضروري لأن pg_xlog قد يحتوي على سجلات لم يتم وضعها في الأرشيف قبل انقطاع النظام:

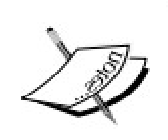
```
mv $PGDATA /tmp
```

٣. بعد ذلك، قم باستعادة ملفات قاعدة البيانات من النسخ الاحتياطي من نظام الملفات الخاص بك، الذي قمت به في وقت سابق. يرجى التأكد من أن تتم عملية الترميم باستخدام الملكية الصحيحة والأذونات المناسبة. إذا كنت تستخدم مساحات الجداول، يجب التأكد من استعادة الروابط الرمزية في دليل pg_tblspc بشكل صحيح:

```
tar -xvf /home/abcd/pgsql/backup.tar
```

٤. قم بإزالة أي ملفات موجودة من دليل pg_xlog، حيث يبدو أنها تأتي من النسخ الاحتياطي لنظام الملفات وربما تكون قد تجاوزها الزمن. إذا لم يتم أرشفة pg_xlog / في وقت سابق، فستحتاج إلى إعادة إنشائه باستخدام الأذونات المناسبة، مع الحرص على التأكد من إعادة إنشائه كرابط رمزي، إذا تم إعداده بهذه الطريقة من قبل:

```
rm -rf /home/abcd/pgsql/data/pg_xlog/*
```



إذا كان هناك أي ملفات جزء وال غير المحفوظة التي تم حفظها في الخطوة ٢، ثم تحتاج إلى نسخها إلى pg_xlog / الموقع.

٥. ستكون الخطوة التالية هي ضبط ملف recovery.conf في دليل البيانات. يمكنك نسخ الملف recovery.conf.sample من دليل share، والذي يقع تحت دليل التثبيت postgres، وبمجرد نسخ الملف recovery.conf.sample إلى دليل البيانات، سوف تحتاج إلى إعادة تسمية باسم ملف recovery.conf:

```
cp /home/abcd/pgsql/share/recovery.conf.sample $PGDATA
cd $PGDATA
mv recovery.conf.sample recovery.conf
```

٦. المعلمة الوحيدة التي يجب تضبيطها في ملف recovery.conf هي المعلمة restore_command. تخبر هذه المعلمة بوستجريسكل كيفية استعادة مقاطع ملف التسجيل عبر الكتابة المؤرشفة:

```
restore_command = 'cp /home/abcd/pgsql/archive/%f %p'
```

٧. وبمجرد الانتهاء من ذلك، فأنت على استعداد لبدء الخادم. سيشغل الخادم في وضع الاستعادة وسيعالج كل ملفات WAL المؤرشفة التي يحتاج إليها. بمجرد اكتمال عملية الاستعادة، سيقوم الخادم بإعادة تسمية recovery.conf إلى recovery.done، ومع ذلك، يتم استرداد قاعدة البيانات المستردة وكنت على استعداد لبدء العمليات العادية مقارنة بقاعدة البيانات:

```
pg_ctl -D $PGDATA star
```

فيما يلي مقتطف من السجل:

```
LOG: starting archive recovery
LOG: archive recovery complete
LOG: database system is ready to accept connections
LOG: autovacuum launcher started
```

يمكنك أيضا أن ترى أن الملف recovery.conf يعاد تسميته الآن إلى recovery.done بمجرد اكتمال الاستعادة:

```
[postgres@localhost data]$ ls -ltrh |tail -7
-rw-r--r-- 1 postgres postgres 4.7K Jul 4 04:13 recovery.done
-rw------- 1 postgres postgres 78 Jul 4 04:14 postmaster.pid
-rw------- 1 postgres postgres 59 Jul 4 04:15 postmaster.opts
drwx------ 2 postgres postgres 4.0K Jul 4 04:15 pg_notify
drwx------ 2 postgres postgres 4.0K Jul 4 04:15 global
```

```
drwx------ 3 postgres postgres 4.0K Jul 4 04:15 pg_xlog
drwx------ 2 postgres postgres 4.0K Jul 4 04:24 pg_stat_tmp
```

كيف تعمل...

تعمل استعادة نقطة في الوقت بهذه الطريقة.

أولا، تحتاج إلى استعادة دليل البيانات من ملف النسخ الاحتياطي. قاعدة البيانات في هذه المرحلة في حالة غير متناسقة لأنه تسترد في وقت النسخ الاحتياطي. ولا تزال بحاجة إلى حساب أي تغييرات أجريت من قبل المعاملات التي كانت مستمرة خلال النسخ الاحتياطي. لهذا، تحتاج إلى تطبيق الأرشيفات. تعيين قيمة restore_command في الخطوة ٦ في مقطع "كيف ينجز ذلك..." يضمن أن التغييرات المسجلة في أقسام WAL المؤرشفة يتم تطبيقها على قاعدة البيانات. بمجرد أن يشغل الخادم، فإنه يعمل في وضع الاسترداد من أجل استعادة كافة البيانات بشكل صحيح. بعد بضع دقائق، سيتم استعادة قاعدة البيانات بنجاح إلى نقطة التفتيش الأخيرة التي أرشفت فيها السجلات.

هناك المزيد...

إذا كان شرط العمل هو استعادة قاعدة البيانات إلى نقطة في وقت سابق، سنحتاج إلى تحديد نقطة التوقف المطلوبة في ملف التضبيط recovery.conf. نقطة التوقف، ويشار إليها أيضا باسم هدف الاستعادة، ويمكن تحديدها من حيث البيانات والوقت، و اسم نقطة الاستعادة ، أو عن طريق انهاء معرف معاملة محددة.

المعلمات التي يمكن التحكم في استعادة نقطة في الوقت في ملف recovery.conf هي recovery_target_time و recovery_target_xid. قد تضبط أي من هذه الخيارات للتحكم في استعادة نقطة في وقت سابق.

استعادة قواعد البيانات وكائنات قاعدة بيانات محددة

في هذه الوصفة،سوف نتحدث عن كيفية استعادة قاعدة بيانات واحدة، وجميع قواعد البيانات، وأشياء محددة، مثل الجداول.

كيف ينجز ذلك...

هنا، سوف نتحدث عن ثلاثة سيناريوهات: ما يجب اتباعه عندما تحتاج إلى استعادة كافة قواعد البيانات على الخادم أو قاعدة بيانات محددة أو جدول محدد فقط. وسوف نغطي هذه السيناريوهات في سلسلة من الخطوات، كما هو موضح على النحو التالي:

١. استعادة كافة قواعد البيانات: في النسخ الاحتياطي المنطقي لكافة قواعد البيانات بوستجريسكل، أنشأنا تفريغ منطقي لجميع قواعد البيانات على الخادم. سيستخدم ملف التفريغ، all.sql هنا لاستعادة كافة قواعد البيانات على الخادم ، على افتراض أن ملفات قاعدة البيانات تالفة وقد انهار الخادم. وهذا أمر استعادة كافة قواعد البيانات هنا:

```
psql -U postgres -f c:\pgbackup\all.sql
```

٢. استعادة قاعدة بيانات واحدة: في النسخ الاحتياطي المنطقي من وصفة قاعدة بيانات بوستجريسكل واحدة، أنشأنا نسخة احتياطية لها باستخدام أداة pg_dump وأسمينا تفريغها dvdrental.tar.

الآن، سنقوم باستعادة قاعدة بيانات dvdrental على افتراض أنه قد حذفت بالفعل، على النحو التالي:

```
pg_restore --dbname=dvdrental --create --verbose
/home/abcd/dvdrental.tar
```

الخيار create- من الأمر pg_restore يقوم بإنشاء قاعدة بيانات فارغة قبل استعادة كافة الكائنات من ملف تفريغ dvdrental.tar.

٣. استعادة جدول واحد: يمكننا أيضا استعادة الكائنات الفردية مثل الجداول. هنا، سنقوم بحذف جدول اسمه "متجر" " store" ثم استعادة الجدول باستخدام تفريغ منطقي الذي استخدم لاستعادة قاعدة بيانات dvdrental في الخطوة السابقة:

١. أولا، حذف جدول المتجر:

```
dvdrental#drop table store cascade;
```

٢. في الخطوة التالية، سنقوم باستخراج تعريف الجدول من التفريغ المتاح ثم نقوم بتفريغ الجدول وتعريفه، فضلا عن البيانات التي فيه في ملف جديد:

```
pg_restore -t store dvdrental.tar > droppedtable.sql
```

هنا، الملف droppedtable.sql يحتوي على تعريفات الجدول، جنبا إلى جنب مع البيانات الضرورية لاستعادة جدول المتجر الذي حذف في الخطوة السابقة.

٣. ستكون الخطوة الأخيرة هي استخدام الملف الذي أنشئ حديثا والذي يحتوي على تعريف الجدول والبيانات لاستعادة الجدول في قاعدة بيانات dvdrental:

```
psql -f droppedtable.sql dvdrental
```

كيف تعمل...

الأداة المساعدة pg_restore تمكنك من استعادة قواعد البيانات التي تم الاحتفاظ بنسخة احتياطية لها من قبل pg_dumpall pg_dumpor. فهي أداة لاستعادة قواعد بيانات بوستجريسكل من أرشيف أنشيء بواسطة pg_dump في واحدة من تنسيقات النص غير عادي.

ستقوم الأداة المساعدة pg_restore بإصدار الأوامر اللازمة لاستعادة قاعدة البيانات إلى الحالة التي كانت في وقت حفظها. تسمح ملفات الأرشيف لـ pg_restore أن تكون انتقائية حول ما يتم استعادته.

هناك المزيد...

لتسريع عملية الاستعادة، من الممكن إجراء عمليات استعادة موازية في بوستجريسكل. يستخدم مفتاح التبديل j- لتحديد عدد خيوط العمليات المطلوبة للاستعادة. كل خيط thread يقوم باستعادة جدول منفصل في وقت واحد، مما يؤدي إلى تسريع عملية الاستعادة.

٤- مهام الصيانة الروتينية

في هذا الفصل، سنغطي الوصفات التالية:

- التحكم في صيانة قاعدة البيانات التلقائية
- منع التجميد التلقائي وفساد الصفحة
- منع إخفاقات التفاف و رقم تعريف المعاملة
- تحديث إحصاءات المخطط
- التعامل مع انتفاخ الجداول والفهرسات
- بيانات المراقبة وصفحات الفهرس
- إعادة الفهرسة الروتينية
- الحفاظ على ملفات السجل

مقدمة

من المهم القيام بعمليات الصيانة الدورية على فترات منتظمة لقاعدة بيانات بوستجريسكل لتحقيق الأداء الأمثل. يمكن لمعاملات قواعد البيانات الثقيلة أن تترك وراءها قدرا كبيرا من البيانات، مما قد يؤدي إلى انخفاض في أداء قاعدة البيانات. وبالتالي، يحتاج مدير قاعدة البيانات لتنفيذ عمليات الصيانة من أجل تنظيف قاعدة البيانات وتحسين أداء قاعدة البيانات. في هذا الفصل، سنناقش كيفية التعامل مع انتفاخ الجداول والفهارس، وإخفاقات رقم تعريف المعاملات، ومهام الصيانة، مثل التنظيف.

التحكم في صيانة قاعدة البيانات التلقائية

بوستجريسكل لديه ميزة تعرف باسم التنظيف التلقائي "autovacuum" ، والتي على الرغم من أنها اختيارية، تفعل افتراضيا في الإصدارات الرئيسية لبوستجريسكل، بدءا من بوستجريسكل ٩.٠ فالأحدث. الوظيفة الخفية لـ التنظيف التلقائي هو تنفيذ الأوامر تنظيف "VACUUM" و فحص "ANALYZE" وتنفيذ مهام الصيانةهذه.

كيف ينجز ذلك...

على الرغم من تفعيل التنظيف التلقائي بشكل افتراضي في بوستجريسكل،فقد تحتاج إلى التأكد من التنظيف التلقائي يعمل بالفعل. تفعيل خدمة التنظيف التلقائي يتطلب منك ضبط وتفعيل المعلمات التالية في ملف التضبيط :postgresql.conf

```
autovacuum = on
track_counts = on
```

وكما يوحي اسمها، فإن معلمة التنظيف التلقائي تتحكم بما إذا كان يجب على الخادم إطلاق خدمة التنظيف التلقائي الخفي أم لا.

تمكن معلمة track_counts من جمع الإحصاءات من نشاط قاعدة البيانات. وعادة ما تفعّل هذه المعلمة بشكل افتراضي لأن معظم التدقيقات التي يؤديها التنظيف التلقائي تتطلب استخدام مجموعة الإحصاءات، وما لم تفعل ميزة جمع الإحصاءات، فإنه لا يمكن استخدام التنظيف التلقائي.

الإعداد السابق من تفعيل التنظيف التلقائي يحدث على المستوى العام، كما هو معرف في ملف التضبيط postgresql.conf. ومن الممكن أيضا تفعيل التنظيف التلقائي على مستوى الجدول، على النحو التالي:

```
ALTER TABLE hrms SET (
autovacuum_enabled = TRUE, toast.autovacuum_enabled = TRUE
);
```

هنا قمنا بتفعيل التنظيف التلقائي للجداول TOAST كذلك. عادة، توضع قيم البيانات الطويلة في جدول ثانوي يعرف باسم جدول TOAST. وبالتالي، لكل جدول فعلي، سيكون هناك جدول TOAST في مقابله يحتوي على قيم البيانات الطويلة، وبالتالي سيعرف فهرس TOAST المعني كذلك.

كيف تعمل...

في البداية، التنظيف التلقائي يدقق الجداول المرشحة و المؤهلة للكنس. ويقوم بذلك عن طريق التحقق من الجداول التي تحتوي على عدد كبير من الصفوف التي تم إدراجها أو تحديثها أو حذفها؛ وهي، الصفوف المجزأة. عندما ينتهي

التنظيف التلقائي من التعرف على الجداول المجزأة، يتم تكليف جميع عمال التنظيف التلقائي بمهمة كنس الجداول المجزأة.

بعد أن ناقشنا في وقت سابق المهمة التي يؤديها التنظيف التلقائي، دعونا الآن نناقش عملية التنظيف التلقائي في حد ذاتها. التنظيف التلقائي يتكون أساسا من عمليات متعددة. هناك عملية تنظيف تعمل باستمرار معروفة باسم مطلق التنظيف التلقائي autovacuum launcher ، الذي تتمثل مهمتها في بدء عامل التنظيف التلقائي لجميع قواعد البيانات الموجودة على خادم بوستجريسكل. سوف يحاول جهاز التشغيل التلقائي تشغيل عامل واحد داخل كل قاعدة بيانات بعد انقضاء القيمة المحددة، بالثواني، في المعلمة autovacuum_naptime؛ فإن المشغل سيقوم بتوزيع العمل وفقا لذلك لكل عامل. وظيفة العامل هي العثور على الجداول المجزأة في قاعدة البيانات الخاصة به وتنفيذ الأوامر التنظيف و الفحص عند الحاجة تلقائيا.

لمزيد من المعلومات حول التنظيف التلقائي، يرجى زيارة هذا الرابط

http://www.postgresql.org/docs/9.3/static/runtime-config-autovacuum.html

هناك المزيد...

هناك عدد كبير من المعلمات التنظيف التلقائي التي تتحكم في سلوك ميزة التنظيف التلقائي. وسنناقش بعض هذه المعايير على النحو التالي:

- log_autovacuum_min_duration: تساعد هذه المعلمة على مراقبة النشاط التلقائي. تحدد هذه المعلمة أن كل إجراء يتم تنفيذه بواسطة التنظيف التلقائي يتم تسجيله إذا تم تنفيذه على الأقل في الوقت المحدد بالمللي ثانية في هذه المعلمة.
- autovacuum_max_workers: تحدد هذه المعلمة الحد الأقصى لعدد العمليات التي يمكن تنفيذها في أي وقت معين. عملية مطلق التنظيف التلقائي هو استثناء في هذا؛ وبالتالي، فإنه لا يتم احتسابه أو عدّه في قائمة عمال vacuum max.
- autovacuum_vacuum_threshold: تحدد هذه المعلمة عدد الصفوف المحدثة والمحذوفة لبدء VACUUM في الجدول المتصل به.
- autovacuum_analyze_threshold: تحدد هذه المعلمة عدد الصفوف المحدثة والمحذوفة لبدء ANALYZE في الجدول المتصل به.

- autovacuum_vacuum_scale_factor: تحدد قيمة هذه المعلمة جزء حجم الجدول الذي يجب إضافته إلى autovacuum_vacuum_threshold عند تحديد ما إذا كان سيتم تشغيل VACUUM..
- autovacuum_analyze_scale_factor: تحدد قيمة هذه المعلمة جزء حجم الجدول الذي يجب إضافته إلى autovacuum_vacuum_threshold عند تحديد ما إذا كان سيتم تشغيل ANALYZE.
- autovacuum_freeze_max_age: تحدد قيمة هذه المعلمة القيمة القصوى التي يمكن أن يحققها حقل الجدول pg_class.relfrozenxid قبل إجراء عملية VACUUM لمنع رقم تعريف المعاملة (XID) ملفوفة داخل الجدول. هذه المعلمة تضع حدا إلى أي مدى سوف يتيح التنظيف التلقائي لك الذهاب قبل أن يبدأ بالتدخل ويبدأ عملية تنظيف مكثفة بتجميد كامل XIDs القديمة في الجداول الخاصة بك مع الصفوف القديمة. إنها ليست فكرة جيدة أن ترفع قيمة هذه المعلمة إلى أقصى حد لأنه يمكن أن تولد الكثير من الإدخال/الإخراج عدة ألوف مرات ، مما تسبب في مشاكل الأداء، وتجميد التنظيف التلقائي لا يمكن إلغاؤه.
- autovacuum_vacuum_cost_delay: تحدد هذه المعلمة تكلفة قيمة فترة التأخر التي سيتم استخدامها في عمليات التنظيف .

منع التجميد التلقائي وفساد الصفحة

في بيئة OLTP ، نتوقع عادة، وعادة ما نرى أن هناك الكثير من عمليات معالجة البيانات على الجداول. بسبب عمليات معالجة البيانات المتكررة على الجداول، يمكننا أن نرى الصفوف التي تم حذفها أو التي تجاوزها الزمن بسبب عملية التحديث؛ ومع ذلك، لم تتم إزالتها فعليا من الجداول الخاصة بها. ويشار إلى هذه الصفوف على أنها **سطور ميتة**.

ومن الممكن أيضا أن يكون إصدار الصف قد أصبح كبيرا بما فيه الكفاية ليصبح مرشحا لتجميده. ويشار إلى هذه الصفوف باسم **السطور المجمدة**.

تتعامل التنظيفات مع كل من الصفوف الميتة، من خلال استعادة مساحة من الصفوف الميتة، وإصدارات الصف القديمة، عن طريق تجميدها بحيث تحفظ حتى يأتي وقت حذفها.

كيف ينجز ذلك...

يحدث التجميد عندما يصبح XID، الذي هو رقم تعريف المعاملة، على الصف أكبر من قيمة vacuum_freeze__min_age الأقدم من القيمة الحالية التالية. يتطلب إجراء تدقيق للجدول للتأكد من أن كافة معرفات المعاملات القديمة تستبدل بـ FrozenXID. تتحكم المعلمة vacuum_freeze_table_age عند إجراء فحص

على الجدول بأكمله. إن تعيين قيمة المعلمة vacuum_freeze_table_age إلى صفر تجبر عملية التنظيف أن تعمل بشكل دائم على مسح كافة الصفحات. إن عملية تدقيق كافة الصفحات كتلة كتلة لكامل قاعدة البيانات أثناء تشغيل التنظيف هو أيضا وسيلة فعالة لتأكيد من عدم وجود أعطاب الصفحات. ويمكن الشروع في ذلك على مستوى قاعدة البيانات على النحو التالي:

```
SET vacuum_freeze_table_age = 0;
VACUUM;
```

ويمكن البدء بذلك على مستوى الجدول كما يلي:

```
VACUUM demo;
```

هنا، demo هو اسم الجدول الذي سيجري تنظيفه.

كيف تعمل...

التنظيف يتناول أداء المهام التالية:

- استعادة أو إعادة استخدام مساحة القرص التي تشغلها الصفوف الميتة.
- إبقاء مجموعة الإحصائيات محدثة، والذي يستخدمها مخطط استعلام بوستجريسكل.
- الحماية من خسارة المعاملات القديمة بسبب مشاكل التفاف رقم تعريف المعاملات والذي سيناقش في الوصفة التالية .

إذا لم تكتشف أي أعطاب في الصفحات، فإنه يمكنك حينها أن تستخدم أداة pageinspect لفحص محتويات صفحات قاعدة بيانات على المستوى المنخفض، وهذا أمر مفيد من منظور التنقيح واكتشاف العلل، ويمكن استخدامها أيضا في فحص فهرس الصفحات.

هناك حالتان يمكن أن تنتج عمليات الإدخال/الإخراج ضخمة أثناء التجميد حين بدء تشغيل عملية التنظيف:

- عندما يكون هناك العديد من الصفوف بنفس رقم تعريف المعاملات خلال فترة التجميد.
- عندما تشتغل عملية تدقيق الجدول، وتواجه عددا كبيرا من السطور التي تحتاج إلى تجميد.

منع فشل التفاف رقم تعريف المعاملات

من أجل MVCC ، يستخدم بوستجريسكل رقم تعريف المعاملات الذي هو بطول ٣٢ بت.

وليس من الممكن أن يكون رقم تعريف المعاملة أكبر من ذلك لأن ذلك سيزيد من حجم كل صف كثيرا. إن قيمة ٣٢-بت يمكن أن تستوعب ٤ مليارات من المعاملات، وعلى كل حال، يمكنها أن تتعامل مع نطاق معاملات يصل إلى الملياربن قبل العودة إلى الصفر. وعندما يتجاوز هذا النطاق تظهر المعاملات الماضية الآن وكأنها من المستقبل. أي أن مخرجاتها تصبح غير مرئية، حيث يؤدى ذلك إلى كارثة فقدان البيانات، و لن تعمل قاعدة البيانات بطريقة عقلانية بعد ذلك.

لتفادى فقدان البيانات القديمة. يجب أن تحمل الصفوف القديمة رقم تعريف المعاملات (XID) و

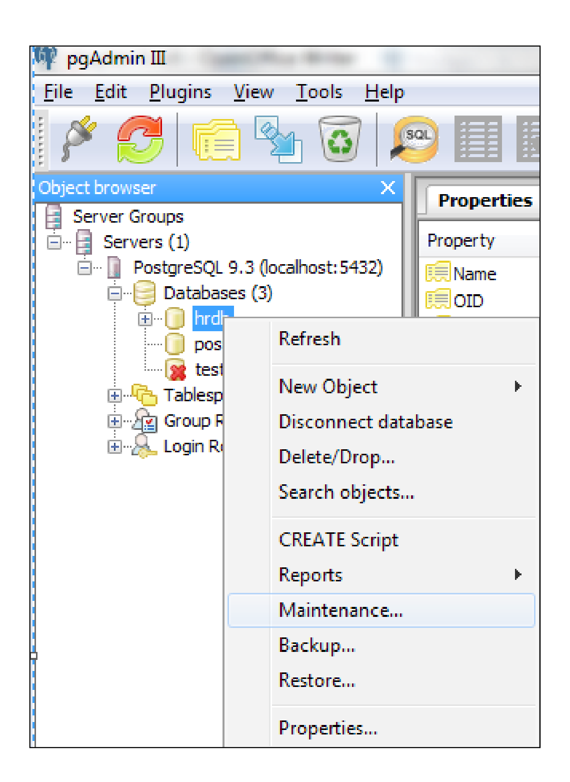
FrozenXID (تجميد رقم تعريف المعاملات) لبعض الوقت قبل أن تصل إلى حد المليارن. وبمجرد أن يسند لهذه الصفوف معرف FrozenXID ، ستظهر على أنها من الماضي لجميع المعاملات العادية بغض النظر عن مشاكل الالتفاف، ستكون هذه الصفوف سيكون جيدة حتى يحين موعد حذفها، مهما طال الزمن لذلك. يقوم الأمر VACUUM بإعادة تعيين XID .

كيف ينجز ذلك...

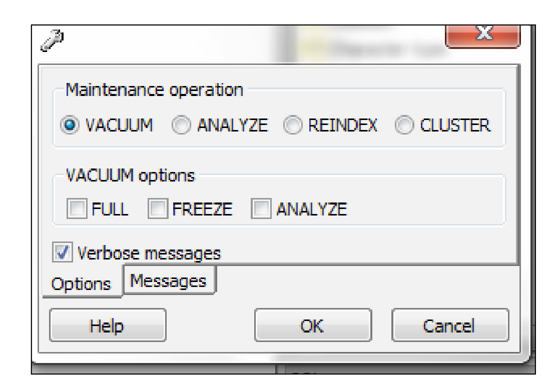
في الوصفة السابقة رأيت استخدام أمر التنظيف على مستوى قاعدة البيانات وكذلك على الجدول. كبديل عن استخدام أمر VACUUM ، يمكنك أيضا استخدام أداة vacuumdb لتنظيف قاعدة البيانات بوستجريسكل. ومثل أمر VACUUM تولد أداة vacuumdb أيضا إحصاءات يستخدمها محسن الاستعلامات. إن أداة vacuumdb هي مجرد التفاف حول الأمر VACUUM. يمكننا استخدام أداة vacuumdb لتنظيف قاعدة بيانات hrdb على النحو التالي:

```
$ vacuumdb hrdb
```

يمكنك أيضا استخدام أداة pgAdmin لتنظيف قاعدة بيانات. لتنظيف قاعدة بيانات في pgAdmin في إطار مستعرض الكائنات Object browser في الجزء الأيسر، انقر بزر الفارة الأيمن فوق قاعدة بيانات التي ترغب فيها من قائمة قواعد البيانات، انقر فوق Maintenance... على النحو المبين في لقطة الشاشة:



يؤدى هذا إلى فتح مربع حوار حيث يمكنك فيه تحديد زر تنظيف VACUUM، ثم انقر فوق موافق من أجل تنظيف قاعدة بيانات, كما هو موضح هنا:



تحديث مخطط الإحصاءات

من أجل إنشاء خطة جيدة للاستعلامات، يعتمد مخطط الاستعلام بوستجريسكل على المعلومات الإحصائية المتاحة حول محتويات الجداول. ولذلك، من الضروري ضمان أن تكون الإحصاءات دقيقة ومحدثة. إذا كانت الإحصاءات قديمة، فإنه سوف يؤدي إلى وضع خطة سيئة للاستعلامات، الأمر الذي سيؤدي إلى مزيد من تدهور أداء قاعدة البيانات.

كيف ينجز ذلك...

هناك طريقتان يمكن بها جمع الإحصاءات:

- يمكنك تشغيل أمر ANALYZE لإنشاء إحصاءات على الجداول.
- يمكن استدعاء أمر ANALYZE كخطوة اختيارية أثناء استخدام التنظيف. إذا فعل التنظيف التلقائي، فإنه سيشغل تلقائيا استدعاء أمر ANALYZE عندما تتغير محتويات الجدول بشكل كبير.

إن تفاصيل التنظيف التلقائي الخفي و أمر VACUUM قد غطيناها في الوصفتين السابقتين، لذا الآن نركز على أمر ANALYZE هنا:

```
ANALYZE demo;
```

في الحالة السابقة, استخدمنا أمر ANALYZE لتوليد الإحصاءات حول الجدول المسمى demo و تخزين النتائج في جدول كتالوج النظام pg_statistic .

جمع الإحصاءات هو أمر في غاية الدقة ويمكن حتى أن يحدث ذلك على مستوى العمود.

كيف تعمل...

تتضمن الإحصائيات التي تجمع بواسطة الأمر ANALYZE بعض أكثر القيم شيوعا في كل عمود جنبا إلى جنب مع رسم بياني يوضح توزيع البيانات التقريبي في كل عمود.

بالنسبة للجداول الكبيرة، وبدلا من فحص كل صف، يأخذ ANALYZE عينة عشوائية من محتويات الجدول. الفائدة من استخدام هذا النهج هو أنه باستخدام طريقة أخذ العينات العشوائية، حتى الجداول الكبيرة يمكن فحصها في فترة قصيرة من الزمن.

يمكن التحكم في كمية الإحصاءات التي جمعت بواسطة ANALYZE عن طريق ضبط قيمة معلمة الضبط default_statistics_target.

يكتسب الأمر ANALYZE قفل للقراءة فقط على الجدول الهدف. وبهذه الطريقة، يمكن تشغيله بالتوازي مع أي نشاط آخر على الجدول المستهدف.

التعامل مع جداول bloating والفهارس

من الشائع العثور على قاعدة بيانات حيث أوقفت عملية التنظيف إما لجدول أو لقاعدة البيانات بأكملها. السبب في إيقاف التنظيف هو أن التنظيف يخلق الكثير من الإدخال/الإخراج في القرص. وهذا قد يساعد مؤقتا، ولكن على المدى الطويل، فمن غير المستحسن أن تقوم بإيقاف التنظيف أو التخلي عن ذلك. من ناحية أخرى، إذا كانت عملية التنظيف تعمل بشكل متكرر، يمكن أن يكون أداء النظام بطيئا لأن التنظيف يخلق الكثير من حركة مرور عمليات الإدخال/الإخراج.

إذا أبقيت قاعدة البيانات دون تنظيف أو إذا كانت البيانات سيئة التنظيم، قد تواجه انتفاخ الجداول والفهارس. المشكلة مع الجداول المنتفخة والفهارس هي أنها تشغل مساحة تخزين أكبر مما هو مطلوب، والذي غالبا ما يسبب مشاكل الأداء عندما تستخدم هذه البيانات من قبل الاستعلامات. في هذه الوصفة، سنتعلم كيفية الكشف عن انتفاخ الجداول والفهارس وما أفضل وقت لتشغيل الأمر تنظيف. إذ أنه كلما كان هناك الكثير من الصفوف الميتة في الجدول، فإن نسبة الانتفاخ تكون أعلى.

كيف ينجز ذلك...

هنا سنتعلم متى يصبح الجدول منتفخا وكيفية التعامل معه:

١- أولا، نقوم بتنشيط وحدة pgstattuple و التي تستخدم لاكتشاف الجدول المنتفخ على النحو التالي:

```
hrdb=# create schema stats;
hrdb=# create extension pgstattuple with schema stats;
```

٢- ثانيا سنقوم بتشكيل جدول و نضيف فيه بعض الصفوف:

```
hrdb=# CREATE TABLE num_test AS SELECT *
FROM generate_series(1, 10000);
```

٣- الآن، سنقوم باستخدام وظيفة pgstattuple المزودة من طرف إضافة pgstattuple

لفحص إحصائيات على مستوى الصفوف لـ num_test :

```
hrdb=# SELECT * FROM stats.pgstattuple('num_test');
-[ RECORD 1 ]------+-------
table_len           | 368640
tuple_count         | 10000
tuple_len           | 280000
tuple_percent       | 75.95
dead_tuple_count    | 0
```

```
dead_tuple_len          | 0
dead_tuple_percent      | 0
free_space              | 7380
free_percent            | 2
```

٤- عند هذه النقطة، لا نرى أي من الصفوف الميتة، إذن سنقوم بحذف بعض البيانات من جدول Num_test:

```
hrdb=# DELETE FROM num_test WHERE generate_series % 2 = 0;
```

٥- الآن سوف نعيد استخدام وحدة pgstattuple لفحص جدول bloat في جدول Num_test :

```
hrdb=# SELECT * FROM stats.pgstattuple('num_test');
-[ RECORD 1 ]------+-------table_len | 368640
tuple_count            | 5000
tuple_len              | 140000
tuple_percent          | 37.98
dead_tuple_count       | 5000
dead_tuple_len         | 140000
dead_tuple_percent     | 37.98
free_space             | 7380
free_percent           | 2
```

في هذا الناتج، يمكنك أن ترى أن نسبة صفوف الميتة حوالى ٣٨ بالمائة. لذا سنقوم بتنظيف الجدول من أجل إزالة انتفاخ الجدول. ولاحظ أيضا أن نسبة مساحة تخزين المتوفرة حوالى ٢ بالمئة, كما يتضح في عمود free_percent:

```
hrdb=# VACUUM num_test;
```

٦- الآن وبعد أن نُظّف الجدول سنقوم بإعادة فحص إحصائيات مستوى-الصف في جدول num_test :

```
hrdb=# SELECT * FROM stats.pgstattuple('num_test');

-[ RECORD 1 ]------+-------
table_len              | 368640
tuple_count            | 5000
tuple_len              | 140000
tuple_percent          | 37.98
dead_tuple_count        | 0
dead_tuple_len          | 0
dead_tuple_percent      | 0
free_space              | 167380
free_percent            | 45.4
```

في الناتج السابق، يمكنك أن ترى أن قيمة عمود dead_tuple_percent تساوي صفر بالمائة, مما يعنى أنه لا يوجد صفوف ميتة. كما زادت مساحة التخزين, ويمكننا أن نرى الآن أن مساحة تخزين متوفرة بحوالي ٤٥ في المائة، مما يدل على أن المزيد من مساحة التخزين أصبحت متاحة بعد التنظيف. قبل التنظيف,المساحة الحرة كانت بنسبة ٢ في المئة تقريبا. وهكذا، بعملية التنظيف، نجحنا في إزالة الانتفاخ الموجود في الجداول.

والآن بعد أن تحدثنا عن الجداول المنتفخة، دعونا ننتقل إلى الفهارس المنتفخة. يساعدنا الاستعلام التالي في تحديد ما إذا كان هناك أي فهارس منتفخة لأي جدول معين:

```
hrdb=# SELECT relname, pg_table_size(oid) as index_size,
100-(stats.pgstatindex(relname)).avg_leaf_density AS bloat_ratio
FROM pg_class WHERE relname ~ 'casedemo' AND relkind = 'i';

Relname                                         | index_size |        bloat_ratio
-------------------------------------------------+------------+-------------
casedemo_inventory_id_idx                        | 507904 |          34.11
casedemo_rental_date_inventory_id_customer_id_idx | 630784 |      26.14
casedemo_pkey                                    | 376832 |          10.25
(3 rows)
```

في الناتج السابق، يمكنك أن ترى نسبة انتفاخ الفهرس bloat_ratio لجميع الفهارس التي تنتمي إلى جدول casedemo.

كيف تعمل...

بوستجريسكل لديه ميزة تعرف باسم MVCC ، و هو التحكم المتزامن للنسخ المتعددة الذي يسمح لك بقراءة البيانات في نفس الوقت الذي يقوم فيه الكُتّاب بالتعديلات. ونظرا لميزة MVCC التي يجري تنفيذها، فنحن غالبا ما نواجه حالات حيث أمر UPDATE يمكن أن يزيد من حجم الجداول والفهارس لأنها تترك وراءها إصدارات الصفوف ميتة. وبالمثل، عمليات الحذف DELETE و الإدراج INSERT تأخذ مساحة يجب استرجاعها عن طريق التنظيف. أيضا، بعض أنماط الحذف يمكن أن يسبب في كتل كبيرة من الفهرس تكون مملوءة بمدخلات فارغة، مما يخلق سيناريو الفهرس المنتفخ bloating-index. للتغلب على مشكلة الفهرس المنتفخ ، تحتاج إلى إعادة بناء الفهارس.سيغطى موضوع إعادة بناء الفهارس في وصفة إعادة الفهرسة الروتينية لاحقا في هذا الفصل.

وبالتالي، من المهم فحص الأعمدة dead_tuple_lenand و dead_tuple_percent من حزمة pgstattuple لجدول معين، وإذا كان كل من هذه الأعمدة لها قيم عالية، فمن الأفضل أن تنظف هذه الجداول في وقت يكون نشاط المعاملات على قاعدة البيانات منخفضا حتى لا تؤثر على أداء قاعدة البيانات.

هناك المزيد...

إذا كنت ترغب في تحديد قيمة الانتفاخ في الجداول والفهارس الخاصة بك، يمكنك استخدام الاستعلام التالي. يستند طلب البحث هذا إلى النص البرمجي check_postgres المتاح في

:http://bucardo.org/wiki/Check_postgres

```
SELECT
  current_database(), schemaname, tablename, /*reltuples::bigint,
relpages::bigint, otta,*/
  ROUND(CASE WHEN otta=0 THEN 0.0 ELSE sml.relpages/otta::numeric END,1) AS
tbloat,
  CASE WHEN relpages < otta THEN 0 ELSE bs*(sml.relpages-otta)::bigint END AS
wastedbytes,
  iname, /*ituples::bigint, ipages::bigint, iotta,*/
  ROUND(CASE WHEN iotta=0 OR ipages=0 THEN 0.0 ELSE ipages/iotta::numeric
END,1) AS ibloat,
  CASE WHEN ipages < iotta THEN 0 ELSE bs*(ipages-iotta) END AS wastedibytes
FROM (
  SELECT
    schemaname, tablename, cc.reltuples, cc.relpages, bs,
    CEIL((cc.reltuples*((datahdr+ma-
      (CASE WHEN datahdr%ma=0 THEN ma ELSE datahdr%ma END))+nullhdr2+4))/(bs-
20::float)) AS otta,
    COALESCE(c2.relname,'?') AS iname, COALESCE(c2.reltuples,0) AS ituples,
COALESCE(c2.relpages,0) AS ipages,
    COALESCE(CEIL((c2.reltuples*(datahdr-12))/(bs-20::float)),0) AS iotta --
very rough approximation, assumes all cols
  FROM (
    SELECT
      ma,bs,schemaname,tablename,
      (datawidth+(hdr+ma-(case when hdr%ma=0 THEN ma ELSE hdr%ma
END)))::numeric AS datahdr,
      (maxfracsum*(nullhdr+ma-(case when nullhdr%ma=0 THEN ma ELSE nullhdr%ma
END))) AS nullhdr2
    FROM (
      SELECT
        schemaname, tablename, hdr, ma, bs,
        SUM((1-null_frac)*avg_width) AS datawidth,
        MAX(null_frac) AS maxfracsum,
        hdr+(
          SELECT 1+count(*)/8
          FROM pg_stats s2
```

```
          WHERE null_frac<>0 AND s2.schemaname = s.schemaname AND s2.tablename
= s.tablename
        ) AS nullhdr
      FROM pg_stats s, (
        SELECT
          (SELECT current_setting('block_size')::numeric) AS bs,
          CASE WHEN substring(v,12,3) IN ('8.0','8.1','8.2') THEN 27 ELSE 23
END AS hdr,
          CASE WHEN v ~ 'mingw32' THEN 8 ELSE 4 END AS ma
        FROM (SELECT version() AS v) AS foo
      ) AS constants
      GROUP BY 1,2,3,4,5
    ) AS foo
  ) AS rs
  JOIN pg_class cc ON cc.relname = rs.tablename
  JOIN pg_namespace nn ON cc.relnamespace = nn.oid AND nn.nspname =
rs.schemaname AND nn.nspname <> 'information_schema'
  LEFT JOIN pg_index i ON indrelid = cc.oid
  LEFT JOIN pg_class c2 ON c2.oid = i.indexrelid
) AS sml
ORDER BY wastedbytes DESC;
```

بيانات المراقبة وصفحات الفهرس

في الوصفات السابقة، رأيت أن التحديثات المتكررة للبيانات تؤدي إلى ظهور صفوف ميتة عبر كل من الجداول والفهارس. هذه الصفوف الميتة تستهلك مساحة التخزين. وبالتالي، فمن المهم مراقبة الجداول والفهارس من أجل تحديد كمية الانتفاخ bloat موجودة في هذه الكائنات.

وبصرف النظر عن الانتفاخ ، هناك جوانب أخرى من الجدول والفهرس التي تحتاج إلى مراقبة.على سبيل المثال، إذا كان هناك أي فهارس غير مستخدمة، فينبغي تحديدها وإزالتها. وبالتالي، تحتاج إلى مراقبة الفهارس غير المستخدمة أيضا.

كيف ينجز ذلك...

م.ق.ب يتطلب عادة بعض المعلومات الإحصائية حول الجداول المخزنة في قاعدة بيانات بوستجريسكل على النحو التالي:

- معلومات عن العدد الإجمالي للصفوف في الجدول و طول الجدول
- معلومات عن عدد الصفوف الميتة ونسبة الصف الميت في الجدول المعني.

- معلومات عن مقدار ونسبة المساحة المتوفرة في الجدول
- المعلومات المتعلقة بعدد التحديثات، وعمليات الإدراج والحذف في الجدول .
- معلومات عن آخر مرة نظفت فيها الجدول، يدويا أو عبر التنظيف التلقائي autovacuum و عن آخر مرة فحصت فيها الجدول لجمع الإحصائيات.

المعلومات الإحصائية السابقة التي يمكن الحصول عليها من وحدة pgstattuple الذي يوفر إحصائيات على مستوى الصف في الجدول المعطى و pg_stat_all_tables التي تبين الإحصاءات عن مرات الوصول إلى جدول محددة.

في الناتج التالي من وحدة pgstattuple يمكنك رؤية عدد الصفوف، و طول الصف، وعدد الصفوف الميتة، ونسبة المساحة الحرة المتاحة ومقدارها:

```
hrdb=# SELECT * FROM stats.pgstattuple('casedemo');
-[ RECORD 1 ]------+--------
table_len             | 1228800
tuple_count           | 16044
tuple_len             | 1152240
tuple_percent         | 93.77
dead_tuple_count      | 0
dead_tuple_len        | 0
dead_tuple_percent    | 0
free_space            | 8184
free_percent          | 0.67
```

يمكنك أيضا استخدام الجدول pg_stat_all_tables لسحب بعض التفاصيل المثيرة للاهتمام، مثل عدد الصفوف التي حدثت وحذفت وأدرجت بالإضافة إلى الطابع الزمني لآخر مرة اكتمل فيها التنظيف التلقائي لهذا الجدول ، والطابع الزمني لآخر مرة حلل الجدول تلقائيا ، وما إلى ذلك:

```
hrdb=# SELECT schemaname,relname,n_tup_ins,n_tup_upd,n_tup_del,
n_live_tup,n_dead_tup,last_autovacuum,last_analyze
from pg_stat_all_tables where relname='casedemo';
-[ RECORD 1 ]---+------------------------------
schemaname            | public
relname               | casedemo
n_tup_ins             | 16044
n_tup_upd             | 0
n_tup_del             | 0
n_live_tup            | 16044
n_dead_tup            | 0
```

```
last_autovacuum         |
last_analyze            | 2014-07-13 20:21:11.636+05:30
```

وتشمل المعلومات الإحصائية المتعلقة بالفهارس كما يلي:

- معلومات عن الأماكن التي تشغلها الفهارس وما إذا كانت هناك أية فهارس منتفخة.
- معلومات حول عدد المرات التي استخدم فيها الفهرس من قبل مخطط الاستعلام.
- معلومات حول عدد الصفوف التي يقرأها الفهرس.
- معلومات حول عدد الصفوف التي جلبت بواسطة الفهرس.
- المعلومات المتعلقة بتجزئة الورقة في الفهرس. تحدث تجزئة الورقة عندما تحذف الصفوف، وبالتالي إنشاء كتل فارغة جزئيا أو كليا تماما في الشجرة الثنائية للفهرس. بسبب حذف الصف، فصفحات مستوى- ورقة الفهرس لها مساحة حرة. ونتيجة لذلك، يستخدم الفهرس المزيد من صفحات البيانات لتخزين البيانات على القرص وفي الذاكرة، مما يؤثر على أداء عمليات التدقيق حتى عندما تخزن صفحات البيانات مؤقتا بسبب صفحات إضافية تحتاج إلى معالجة.

يمكن استرجاع إحصائيات حول الفهارس من الجداول pg_stat_user_indexes و pg_index. يمكنك استخدام هذين الجدولين للعثور على الفهارس غير المستخدمة:

```
SELECT
    relid::regclass AS table,
    indexrelid::regclass AS index,
    pg_size_pretty(pg_relation_size(indexrelid::regclass)) AS index_size,
    idx_tup_read,
    idx_tup_fetch,
    idx_scan
FROM pg_stat_user_indexes
  JOIN pg_index USING (indexrelid)
 WHERE idx_scan = 0
AND indisunique IS FALSE;
```

في ناتج الاستعلام السابق، تشير الأعمدة idx_tup_read و idx_tup_fetch و idx_scan إلى استخدام الفهرس:

- idx_tup_read: يشير هذا العمود إلى عدد الصفوف التي قرئت باستخدام الفهرس
- idx_tup_fetch: يشير هذا العمود إلى عدد الصفوف التي جلبت باستخدام الفهرس
- idx_scan: يشير هذا العمود إلى عدد مرات استخدام الفهرس بواسطة مخطط الاستعلام.

إعادة الفهرسة الروتينية

في بعض السيناريوهات، فإنه يستحق إعادة بناء الفهارس بشكل دوري باستخدام أمر إعادة الفهرسة REINDEX. يمكن أن تصبح الفهارس مشكلة في تطبيقات قاعدة البيانات التي تنطوي على نسبة عالية من الإدخالات والحذف المتكررة، وهذا قد يتسبب في أن تصبح الفهارس منتفخة. إن احتمال حدوث الانتفاخ ليس إلى أجل غير مسمى. وهذا هو، في أسوأ الأحوال سيكون هناك مفتاح واحد لكل صفحة، ولكن قد يكون من المفيد جدولة إعادة النسخ الدورية للفهارس التي لديها أنماط استخدام من هذا القبيل. وبمساعدة الأمر إعادة الفهرسة REINDEX ، تسترجع صفحات الفهرس التي أصبحت فارغة تماما لإعادة استخدامها.

كيف ينجز ذلك...

يمكن إعادة بناء الفهارس على مستويات مختلفة، على النحو التالي:

- يمكنك إعادة إنشاء الفهرس على مستوى الفهرس الفردي، حيث يمكن إعادة إنشاء فهرس واحد. يمكنك إعادة إنشاء فهرس واحد، كما هو موضح هنا:

```
REINDEX INDEX customer_pkey;
```

- يمكنك إعادة إنشاء الفهارس على مستوى الجدول، حيث يعاد بناء كافة الفهارس للجدول المعطى:

هنا، في التعليمات البرمجية التالية، نحن نعيد بناء كافة الفهارس لجدول العملاء:

```
REINDEX TABLE CUSTOMER;
```

- يمكنك إعادة إنشاء الفهارس على مستوى النظام، حيث يمكنك إعادة إنشاء كافة الفهارس على كتالوجات النظام ضمن قاعدة البيانات الحالية. هنا، نحن نعيد بناء كل الفهارس على كتالوج النظام لقاعدة البيانات hrdb:

```
REINDEX SYSTEM hrdb;
```

- يمكنك إعادة إنشاء الفهارس على مستوى قاعدة البيانات، حيث يمكنك إعادة إنشاء كافة الفهارس ضمن قاعدة البيانات الحالية، وهي hrdb في التعليمات البرمجية:

```
REINDEX DATABASE hrdb;
```

كيف تعمل...

يستخدم الأمر إعادة الفهرسة REINDEX لإعادة بناء فهرس باستخدام البيانات المخزنة في جدول الفهرس، وبالتالي استبدال النسخة القديمة من الفهرس.

يستخدم إعادة الفهرسة في الحالات التالية:

- يجب استخدام إعادة الفهرسة REINDEX عندما يصبح الفهرس معطوبا ولا يحتوي على أية بيانات صحيحة. يمكن أن تصبح الفهارس معطوبة بسبب أخطاء البرمجيات أو فشل الأجهزة. يوفر REINDEX طريقة للاسترداد تلك الفهارس.
- إعادة الفهرسة REINDEX تستخدم عندما يصبح الفهرس منتفخا، عندما يحتوي على العديد من الصفحات الفارغة. إعادة الفهرسة تقلل من استهلاك الفضاء في الفهرس عن طريق كتابة نسخة جديدة من الفهرس دون صفحات ميتة.
- إعادة الفهرسة REINDEX تحتاج إلى استخدامها عندما تغير معلمة التخزين للفهرس وترغب في التأكد من أن التغييرات نافذة المفعول.
- إعادة الفهرسة REINDEX يقفل عمليات الكتابة على الجدول الرئيسي للفهرس ولكن لا يمنع عمليات القراءة على الجدول. كما تكتسب إعادة الفهرسة قفل حصري على فهرس معين يجري معالجته، والتي سوف يتم حظر القراءات التي تحاول استخدام الفهرس..

هناك المزيد...

هناك خيار يمكن من خلاله لأمر إعادة الفهرسة REINDEX إعادة بناء فهرس دون إغلاق عمليات الكتابة على الجدول الرئيسي للفهرس. لهذا، يمكنك استخدام الأمر إنشاء فهرس متزامن CREATE INDEX CONCURRENTLY ، والذي سيقوم بإنشاء الفهرس دون اتخاذ أية أقفال تمنع الإدخالات المتزامنة والتحديثات والحذف على الجدول. لذلك، بدلا من إعادة بناء الفهرس، يجب عليك تنفيذ الخطوات الثلاث التالية:

١. أولا، أنشئ فهرسا متطابقا مع الفهرس الذي تريد إعادة إنشائه باستخدام إنشاء فهرس متزامن CREATE INDEX CONCURRENTLY.

٢. بعد ذلك، احذف الفهرس القديم.

٣. والخطوة الأخيرة هي إعادة تسمية الفهرس الجديد بنفس اسم الفهرس القديم.

توضح التعليمات البرمجية التالية الخطوات السابقة:

```
CREATE INDEX CONCURRENTLY card_index ON creditcard (cardno);
BEGIN;
DROP INDEX credit_card_idx;
ALTER INDEX card_index RENAME TO credit_card_idx;
COMMIT;
```

الحفاظ على ملفات السجل

المعلومات المخزنة في ملفات السجل لا تقدر بثمن عند تشخيص أو استكشاف الأخطاء وإصلاح المشاكل. وبمساعدة المعلومات المخزنة في ملفات السجل، يمكنك تحديد مصادر المشاكل في قاعدة البيانات الأساسية. ولهذا السبب، من المهم الحفاظ على ملفات السجل بدلا من التخلص منها. ومع ذلك، تميل المعلومات في ملفات السجل إلى أن تكون ضخمة، لذلك من المهم أن تطبق آلية تدوير من أجل الحفاظ على بعض ملفات السجل وتجاهل ملفات السجل التي لم تعد مهمة. تحتاج ملفات السجل إلى أن تكون بنظام التدوير بحيث حين تبدأ ملفات سجل جديدة تزال القديمة منها بعد فترة معقولة من الزمن.

كيف ينجز ذلك...

هناك آليات مختلفة يمكن من خلالها الاحتفاظ بمعلومات التسجيل وحفظها في ملفات السجل. وتناقش هذه المسائل على النحو التالي:

- من بين الطرق للتعامل مع هذا هو إرسال إخراج stderr الخادم إلى نوع من برامج تدوير السجل. بوستجريسكل لديه تدوير سجل مضمنة فيه ، والتي يمكن استخدامها عن طريق تعيين معلمة الضبط logging_collector في ملف postgresql.conf:

```
logging_collector=true
```

- نهج آخر هو استخدام برنامج تدوير السجل الخارجي الذي قد تستخدمه مع بعض برامج الخادم الأخرى. على سبيل المثال، تتضمن توزيعة أباتشي أداة تعرف باسم rotatelogs التي يمكن استخدامها مع بوستجريسكل. ويمكن القيام بذلك عن طريق الأنابيب إخراج stderr من الخادم إلى البرنامج الخارجي المطلوب. إذا كنت

تستخدم الأمر pg_ctl لبدء خادم بوستجريسكل، فإن مخرجات stderr يعاد توجيهها بالفعل إلى الإخراج، لذلك تحتاج فقط أمر الأنبوب "|"، كما هو موضح هنا:

```
pg_ctl start | rotatelogs /var/log/pgsql_log 86400
```

- النهج الثالث لإدارة إخراج ملف السجل هو إرسال سجل الإخراج إلى syslog والسماح لـ syslog يقوم بمهمة تدوير الملف. للقيام بذلك، تحتاج إلى تعيين المعلمة log_destination إلى سجل النظام syslog في ملف postgresql.conf.

٥- مراقبة النظام باستخدام أدوات يونكس

في هذا الفصل، سنغطي الوصفات التالية:

- مراقبة استخدام وحدة المعالجة المركزية CPU
- مراقبة الترحيل والتبادل
- العثور على أسوأ مستخدم على النظام
- تحميل نظام المراقبة
- تحديد الاختناقات في وحدة المعالجة المركزية
- تحديد الاختناقات في إدخال / إخراج القرص
- أداء نظام المراقبة
- فحص تحميل سجل وحدة المعالجة المركزية
- فحص تحميل سجل الذاكرة
- مراقبة استخدام مساحة القرص
- مراقبة حالة الشبكة

مقدمة

من أجل أن نتمكن من حل مشاكل الأداء، يجب أن نكون قادرين على استخدام أدوات نظام التشغيل بشكل فعال. كما ينبغي أن نكون قادرين على استخدام أدوات وأوامر نظام التشغيل الصحيحة لتحديد مشاكل الأداء التي قد تكون بسبب مشاكل وحدة المعالجة المركزية أو الذاكرة أوفي إدخال / إخراج القرص. وقد تتداخل مهام مدير قاعدة البيانات في كثير من الأحيان مع بعض وظائف إدارة النظام ذات الصلة، ومن المهم أن يكون مدير قاعدة البيانات ذا قدرة فعالة في استخدام الأدوات ذات الصلة في نظام التشغيل من أجل التحديد الدقيق لمكان المشكلة على الخادم. في هذا الفصل،

سنناقش مختلف أدوات نظام تشغيل يونكس / لينكس التي يمكن أن تساعد مدير قاعدة البيانات في تحليل الأداء وقضايا استكشاف الأخطاء وإصلاحها.

مراقبة استخدام وحدة المعالجة المركزية CPU

في هذه الوصفة، سنقوم باستعمال الأمر sar لمراقبة استخدام وحدة المعالجة المركزية على النظام.

الاستعداد للعمل

نفذت الأوامر المستخدمة في هذه الوصفة على جهاز أوراكل سولاريس Oracle Solaris. وبالتالي، قد يختلف ناتج الخرج على باختلاف أنظمة يونكس و لينكس.

كيف ينجز ذلك...

يمكننا استخدام الأمر sar مع المفتاح U- لمراقبة استخدام وحدة المعالجة المركزية:

```
bash-3.2$ sar -u 10 8
SunOS usstlz-pinfsi09 5.10 Generic_150400-04 sun4u    08/06/2014

23:32:17    %usr    %sys    %wio   %idle
23:32:27      80      14       3       3
23:32:37      70      14      12       4
23:32:47      72      13      21       4
23:32:57      76      14       6       3
23:33:07      73      10      13       4
23:33:17      71       8      17       4
23:33:27      67       9      20       4
23:33:37      69      10      17       4

Average       73      11      13       4
```

في الأمر السابق ومع المفتاح u-، مررت قيمتان كمدخلات. القيمة الأولى، وهي ١٠، وهي تعرض عدد الثواني بين قراءات sar، والقيمة الثانية، وهي ٨، و تشير إلى عدد المرات التي تريد تشغيل sar فيها.

كيف ينجز ذلك...

يوفر الأمر sar لمحة سريعة لمدى تعمق أو مدى استخدام وحدة المعالجة المركزية . وتكون قيم تقارير إخراج sar حسب الأعمدة التالية:

- %usr: يشير هذا إلى النسبة المئوية لوحدة المعالجة المركزية التي تعمل في وضع المستخدم
- %sys: يشير هذا إلى النسبة المئوية لوحدة المعالجة المركزية التي تعمل في وضع النظام
- %wix: يشير هذا إلى النسبة المئوية لوحدة المعالجة المركزية التي تعمل بشكل خامل، مع وجود عملية في انتظار الإدخال/الإخراج للكتلة.
- %idle: يشير هذا إلى النسبة المئوية لوحدة المعالجة المركزية الخاملة.

في كثير من الأحيان، النسبة المئوية المنخفضة من وقت الخمول تشير إلى ذروة عمل وحدة المعالجة المركزية أو وحدة المعالجة المركزية التي تعمل بطاقة ضعيفة.

يمكنك استخدام الأمر ps أو الأمر prf في سولاريس لإيجاد وظيفة وحدة المعالجة المركزية المجهدة.

فيما يلي المؤشرات العامة لمشاكل الأداء:

- إذا رأيت قيمه عالية بشكل غير طبيعي في العمود %usr، فهذا يعني أن التطبيقات لم تضبط بطريقه صحيحه أو أنها تستخدم وحدة المعالجة المركزية على نحو زائد.
- إذا كنت تري قيمه عالية في العمود %sys، فانه يشير على الأرجح إلى اختناق يمكن أن يكون سببه التبادل أو الترحيل ويحتاج إلى مزيد من البحث.

مراقبة عمليات الترحيل والتبادل

في هذه الوصفة، سنقوم باستخدام أوامر sar و vmstat مع خيارات لمراقبة عمليات الترحيل والتبادل.

الاستعداد للعمل

من الضروري مراقبة كمية الترحيل والتبادل التي تحدث على نظام التشغيل. حيث يحدث **الترحيل** عندما ينقل جزء من عمليات نظام التشغيل من الذاكرة الفعلية إلى القرص أو تقرأ من الذاكرة الفعلية إلى القرص. ويحدث **التبادل** عند نقل عملية بأكملها إلى القرص من الذاكرة الفعلية أو تقرأ إلى الذاكرة الفعلية من القرص. واستنادا إلى النظام فكل من عمليتي الترحيل أو التبادل يمكن أن تحدث بها مشكلة. إذا حدث الترحيل بشكل طبيعي ولاحظت نوعا من التبادل الثقيل،

فيمكن أن تكون المشكلة متعلقة بعدم كفاية الذاكرة، و في بعض الأحيان يمكن أن تكون متعلقة بحالة القرص أيضا. إذا كان النظام يميل إلى الترحيل بشكل كبير وليس إلى التبادل، فيمكن أن تكون المشكلة متعلقة إما بوحدة المعالجة المركزية أو بالذاكرة. الأوامر المنفذة في هذا القسم من بيئة أوراكل سولاريس.

كيف ينجز ذلك...

يمكننا استخدام أوامر sar و vmstat مع الخيارات على النحو التالي لمراقبة عمليات الترحيل وتبادل البيانات:

١. يمكن استخدام الأمر vmstat مع المفتاح S- لمراقبة التبادل وعمليات الترحيل، على النحو التالي:

```
bash$ vmstat —S

 kthr      memory            page            disk          faults      cpu
 r b w   swap        free      si  so    pi  po  fr de sr s0  s2 s3 s4   in      sy       cs      us   sy   id
 6 14 0  453537392 170151696 0  0    2444 186 183 0 0  1   1   1   1 77696  687853  72596  13   4    83
```

في الأوامر المذكورة أعلاه ، تمثل الأعمدة si و so عمليات التبادل الداخلية والتبادل الخارجية على التوالي.

وبالمثل ، يمثل pi و po عمليات الترحيل الداخلي والترحيل الخارجي على التوالي.

ومع ذلك، فالأمر sar يوفر تحليلا أكثر تعمقا لعمليات الترحيل والتبادل عند استخدامه مع الخيارات.

٢. يمكننا أيضا استخدام الأمر sar مع التبديل p- للإبلاغ عن عمليات ترحيل الصفحات، كما يلي:

```
bash-3.2$  sar -p 5 4

SunOS usmtnz-sinfsi17 5.10 Generic_150400-04 sun4u    08/08/2014

05:45:18  atch/s  pgin/s ppgin/s  pflt/s  vflt/s slock/s
05:45:23 4391.18    0.80    2.20 12019.44 30956.92    0.60
05:45:28 2172.26    1.80    2.40 5417.76 15499.80    0.20
05:45:33 2765.60    0.20    0.20 9893.20 20556.60    0.00
05:45:38 2194.80    2.00    2.00 7494.80 19018.60    0.00

Average  2879.85    1.20    1.70 8703.00 21500.25    0.20
```

تعرض المخرجات السابقة الأعمدة التالية:

- atch/s: هذه هي أخطاء الصفحة المسجلة كل في ثانية التي تتكون عن طريق استعادة صفحة موجودة حاليا في الذاكرة.

- pgin/s: عدد المرات المسجلة في كل ثانية التي يستقبل فيها نظام الملفات الصفحة في الطلبات.
- ppgin/s: هذه الصفحات مقسمة إلى صفحات المسجلة في الثانية.
- pflt/s: هذه هي نسبة أخطاء الصفحات من أخطاء الحماية.
- vflt/s: وهذا هو الرقم الذي يعالج أخطاء صفحه الترجمة المسجلة في كل ثانية. يحدث هذا عند عملية إدخال جدول غير موجود من أجل عنوان ظاهري معين.
- slock/s: هذه هي الأخطاء المسجلة في كل ثانية والتي تسببها طلبات قفل البرمجيات التي تتطلب الإدخال/الإخراج المادي.

٣. وبالمثل، يمكننا استخدام الأمر sar مع المفتاح w- للإبلاغ عن أنشطة التبادل وتحديد ما إذا كان هناك أي مشاكل متعلقة بالتبادل:

```
bash-3.2$ sar -w 5 4

SunOS usmtnz-sinfsi17 5.10 Generic_150400-04 sun4u    08/08/2014

06:20:55 swpin/s bswin/s swpot/s bswot/s pswch/s
06:21:00    0.00     0.0    0.00     0.0   53143
06:21:05    0.00     0.0    0.00     0.0   60949
06:21:10    0.00     0.0    0.00     0.0   55149
06:21:15    0.00     0.0    0.00     0.0   64075

Average     0.00     0.0    0.00     0.0   58349
```

ويعطي الإخراج المذكور أعلاه الأعمدة التالية:

- swpin/s: يشير هذا إلى عدد العمليات خفيفة الوزن التي تتم في الذاكرة في كل ثانية.
- bswin/s: يشير هذا إلى عدد الكتل التي نقلت لوحدات التبديل الإضافية في كل الثانية.
- swpot/s: يشير هذا التقرير إلى متوسط عدد العمليات التي تتبادل خارج الذاكرة في كل ثانية.
- bswot/s: يقوم هذا التقرير بإبلاغ عدد الكتل التي تنقل لمبادلتها في الثانية الواحدة
- pswch/s: يشير هذا إلى عدد محولات خيوط النواة في كل ثانية.

كيف تعمل...

إذا كانت الأعمدة si و so من إخراج vmstat -S لها قيم غير معدومة، فهذا يعتبر بمثابة علامة على احتمال مشكلة في الأداء المتعلق بعملية التبادل. كما يجب إجراء مزيد من الدراسات باستخدام التحليل الأكثر تفصيلا الذي يوفره الأمر sar بمفاتيح p- و w- على التوالي.

بالنسبة إلى الترحيل، فإن النقطة الرئيسية هو البحث عن عدد كبير من أخطاء الصفحة من أي نوع كانت. وهذا من شأنه أن يشير إلى درجة عالية من الترحيل. ومصدر القلق هنا ليس الترحيل وإنما التبادل لأنه كلما زاد الترحيل زاد التبادل. يمكننا أن ننظر إلى القيم في الأعمدة atch/s و pflt/s و vflt/s و slock/s من إخراج الأمر sar -p لمراجعة عدد أخطاء الصفحة مهما كان نوعها، ومشاهدة إحصائيات الترحيل لمراقبة ما إذا كان نشاط الترحيل ثابتا أو يزداد أثناء فتره زمنيه معينه.

لخرج الأمر sar -w ، فإن العمود الرئيسي للمراقبة هو العمود swpot/s.

يشير هذا العمود إلى متوسط عدد العمليات التي تتبادل خارج من الذاكرة في الثانية الواحدة. إذا كانت القيمة في هذا العمود أكبر من ١، فإنه مؤشر على نقص الذاكرة، ولتصحيح هذا سيكون عليك زيادة الذاكرة.

العثور على أسوأ مستخدم على النظام

في هذه الوصفة، سنقوم باستخدام الأمر top للعثور على أسوأ مستخدم على النظام في الأداء في نقطة معينة من الزمن.

الاستعداد للعمل

الأمر top هو أداة تستند إلى لينكس التي تعمل أيضا على الأنظمة المستندة إلى يونيكس. وقد نفذت الأوامر في هذا القسم على جهاز CentOS لينكس.

كيف ينجز ذلك...

يظهر استخدام الأمر top كما يلي:

```
bash-3.2$top

Cpu states: 0.0% idle, 82.0% user, 18.7% kernel, 0.8% wait, 0.5% swap
Memory: 795M real, 12M free, 318M swap, 1586M free swap
PID USERNAME PRI NICE SIZE RES STATE TIME WCPU CPU COMMAND
23624 postgres -25 2 208M 4980K cpu 1:20 22.47% 94.43% postgres
15811 root -15 4 2372K 716K sleep 22:19 0.61% 3.81% java
20435 admin 33 0 207M 2340K sleep 2:47 0.23% 1.14% postgres
20440 admin 33 0 93M 2300K sleep 2:28 0.23% 1.14% postgres
23698 root 33 0 2052K 1584K cpu 0:00 0.23% 0.95% top
23621 admin 27 2 5080K 3420K sleep 0:17 1.59% 0.38% postgres
23544 root 27 2 2288K 1500K sleep 0:01 0.06% 0.38% br2.1.adm
15855 root 21 4 6160K 2416K sleep 2:05 0.04% 0.19% proctool
897 root 34 0 8140K 1620K sleep 55:46 0.00% 0.00% Xsun
20855 admin -9 2 7856K 2748K sleep 7:14 0.67% 0.00% PSRUN
208534 admin -8 2 208M 4664K sleep 4:21 0.52% 0.00% postgres
755 admin 23 0 3844K 1756K sleep 2:56 0.00% 0.00% postgres
2788 root 28 0 1512K 736K sleep 1:03 0.00% 0.00% lpNet
18598 root 14 10 2232K 1136K sleep 0:56 0.00% 0.00% xlock
1 root 33 0 412K 100K sleep 0:55 0.00% 0.00% ini
```

يعطي السطران الأولان في الإخراج السابق معلومات النظام العامة، في حين ترتب بقية الصفوف بناء على استخدام وحدة المعالجة المركزية الحالية.

كيف تعمل...

يوفر الأمر top إحصائيات حول نشاط وحدة المعالجة المركزية (cpu). وهو يعرض قائمة من المهام المجهدة لوحدة المعالجة المركزية على النظام ويوفر أيضا واجهة لمعالجة العمليات.

في الإخراج السابق، يمكننا أن نرى أن المستخدم الأعلى هو postgres ومعرفه بـ ٢٣٦٢٤.

يمكننا أن نرى استهلاك وحدة المعالجة المركزية لهذا المستخدم وهي ٩٤.٤٣ ٪ ، والتي تعتبر مرتفعة جدا وتحتاج إلى تحقيق ، أو أن عملية نظام التشغيل المقابلة تحتاج إلى إيقاف إذا كانت تسبب مشاكل في الأداء على النظام.

مراقبة حمل النظام

في هذه الوصفة ، سنقوم باستخدام الأمر uptime لمراقبة حمل النظام الشامل.

كيف ينجز ذلك...

يعطينا الأمر uptime المعلومات التالية:

- وقت النظام الحالي
- طول المدة التي ظل فيها النظام عاملا.
- عدد المستخدمين المسجلين حاليا في النظام.
- متوسط حمل النظام في فترات ماضية ١، ٥ ، و ١٥ دقيقة.

يمكن استخدام الأمر uptime كما يلي:

```
bash-3.2$ uptime
11:44pm  up 20 day(s), 20 hr(s),  10 users,  load average: 27.80, 30.46, 33.77
```

في الإخراج السابق، يمكننا أن نرى أن وقت النظام الحالي هو ١١:٤٤ (بتوقيت غرينيتش) وأن النظام كان في حالة تشغيل لمدة ٢٠ يوما و ٢٠ ساعة الماضية دون الحاجة إلى إعادة التشغيل.

ويخبرنا الإخراج أيضا أن هناك عشرة مستخدمين مسجلين في نفس الوقت في النظام. وأخيرا، نحصل على متوسط الحمل خلال ١و ٥ و ١٥ دقيقة على التوالي ٢٧.٨٠ و ٣٠.٤٦ و ٣٣.٧٧.

كيف تعمل...

الغرض الأساسي من تشغيل الأمر uptime هو إلقاء نظرة سريعة على حمل وحدة المعالجة المركزية الحالية على النظام. وهذا يوفر نظرة خاطفة على أداء النظام الحالي. ويشير تحميل النظام إلى متوسط عدد أو العمليات التي تكون إما في حالة تشغيل runnable. تدخل العملية في حالة التشغيل عندما تبدأ في استخدام موارد وحدة المعالجة المركزية أو في انتظار الحصول عليها. وتدخل في الحالة المتواصلة uninterruptable عندما يمضي الكثير من الوقت في انتظار عملية إدخال / إخراج. يصنف متوسط حمل على فترات زمنية ثلاث، أي فترات ١، و ٥، و ١٥ دقيقة. ولا تصنف متوسطات الحمل لعدد وحدات المعالجة المركزية على النظام.

ومن ثم , فبالنسبة إلى نظام بوحدة معالجة مركزية واحدة، يشير متوسط الحمل ١ إلى فترة زمنية مشغولة بنسبة ١٠٠ في المائة مع وقت عدم خمول صفري، أما بالنسبة إلى نظام يحتوي على ٥ وحدات معالجة مركزية، فإن متوسط الحمل ١ يشير إلى وقت خمول قدره ٨٠ في المائة وفترة زمنية مزدحمة من ٢٠ في المئة فقط.

تحديد اختناقات وحدة المعالجة المركزية

في هذه الوصفة، سنقوم باستخدام الأمر mpstat لتحديد اختناقات وحدة المعالجة المركزية.

الاستعداد للعمل

نفذت الأوامر في هذا القسم على خادم سولاريس.

كيف ينجز ذلك...

يتم استخدام الأمر mpstat للإبلاغ عن إحصائيات المعالج على شكل جدول.

يظهر استخدام الأمر mpstat كما يلي:

```
bash-3.2$ mpstat 1 1

CPU minf mjf xcal  intr ithr  csw icsw migr smtx  srw syscl  usr sys  wt idl
  0  672   0 2457   681   12  539   17   57  119    0  4303   18  10   0  73

  1   90   0 1551   368   22  344    6   37  104    0  3775   17   4   0  79

  2   68   0 1026   274   14  217    4   24   83    0  2393   11   3   0  86

  3   50   0  568   218    9  128    3   17   56    0  1319    7   2   0  92

  4   27   0  907   340   12  233    3   22   72    0  2034    9   2   0  88

  5   75   0 1777   426   25  370    5   33  111    0  4820   22   4   0  74

  6   69   0 1395   421   15  337    4   27   96    0  2948   14   3   0  83

  7   29   0  888   394    9  273    3   23   74    0  1873    9   3   0  89

  8   10   0  344   177    9   80    2   13   44    0  1007    5   1   0  94

  9   63   0 1275   288   17  268    4   22   90    0  4337   20   3   0  77

 10   72   0 1875   324   28  330    5   30  110    0  5514   25   4   0  71
```

11	17	0	438	183	10	94	2	17	50	0	1048	5	2	0	93
12	10	0	351	175	9	79	2	13	44	0	1047	5	1	0	94
13	60	0	1207	267	17	245	4	21	87	0	4243	19	3	0	78
14	72	0	1859	323	29	331	4	30	109	0	5347	24	4	0	72
15	16	0	434	184	10	94	2	17	50	0	1031	5	2	0	94
16	20	0	638	197	12	127	2	16	57	0	1810	9	2	0	90
17	16	0	621	215	12	141	2	16	59	0	2062	9	2	0	89
18	19	0	785	214	14	151	2	18	63	0	2384	11	2	0	87
19	66	0	1584	293	24	288	4	26	96	0	5681	25	3	0	71
20	23	0	826	207	14	142	2	17	64	0	2172	10	2	0	88
21	14	0	543	190	12	115	2	15	54	0	1896	9	2	0	90
22	19	0	772	210	14	147	2	18	62	0	2347	11	2	0	88
23	66	0	1591	294	24	293	4	26	96	0	5398	25	4	0	72
24	66	0	1820	305	27	311	4	27	100	0	4941	22	4	0	74
25	20	0	672	210	11	135	2	18	62	0	1783	8	2	0	90
26	16	0	645	192	12	116	2	15	59	0	2184	10	2	0	89
27	20	0	821	213	15	152	2	17	62	0	3016	13	2	0	85
28	66	0	1698	305	28	308	4	27	98	0	5106	23	4	0	74
29	20	0	641	194	13	121	2	17	59	0	1731	8	2	0	90
30	17	0	633	190	11	118	2	15	57	0	2164	9	2	0	89
31	22	0	798	215	15	161	2	17	61	0	3044	13	2	0	85

```
 32  637    0 2183    507    21  672    17    61   114     0  4939    20    9    0
70
 33   98    0 1998    383    24  431     7    39    94     0  4076    19    4    0  77

 34   74    0 1217    273    14  265     4    25    73     0  2589    13    3    0  85

 35   54    0  661    216     9  168     3    18    51     0  1624     8    2    0  90

 36   17    0  925    311   117  144     2    18    49     0  1610     8    2    0  90

 37   69    0 2146    302    23  312     4    28    86     0  4624    22    3    0  75

 38   61    0 1856    910   665  254     3    22    71     0  2734    13    5    0  82

 39   12    0 1006    848   661  138     2    15    40     0  1099     5    4    0  91

 40    9    0  402    168     9   82     2    12    32     0   986     5    1    0  94

 41   59    0 1490    288    16  285     4    21    73     0  4233    20    3    0  77

 42   70    0 2486    329    26  356     5    29    91     0  5326    25    4    0  71

 43   16    0  541    180    10   99     2    16    39     0  1052     5    1    0  93

 44   10    0  438    169     8   83     2    13    34     0  1051     5    1    0  94

 45   57    0 1436    264    16  257     3    20    69     0  4137    19    3    0  78
```

كيف تعمل...

في الإخراج السابق من الأمر mpstat، كل صف من الجدول يمثل نشاط معالج واحد. يعرض الجدول الأول ملخصا للنشاط منذ وقت الإقلاع الأخير.

القيمة الهامة من منظور مدير قاعدة البيانات هي القيمة في العمود smtx. . ويشير قياس smtx إلى عدد مرات فشل وحدة المعالجة المركزية في الحصول على تامين الاستبعاد المتبادل (mutex) . حيث يقوم موتكس بتأجيل وقت وحدة المعالجة المركزية المهمل وتقليص القياس متعدد المعالجات.

والقاعدة العامه للتصفح هي أنه إذا كانت القيمة في العمود smtx أكبر من ٢٠٠ ، فإن هذا عبارة عن عرض ومؤشر لحدوث مشاكل اختناق في وحدة المعالجة المركزية ، والتي تحتاج إلى التحقيق فيها.

تحديد الاختناقات في إدخال/إخراج القرص

في هذه الوصفة، سنقوم باستخدام أمر iostat لتحديد الاختناقات المتعلقة بالقرص.

الاستعداد للعمل

نفذت الأوامر في هذا القسم على خادم سولاريس.

كيف ينجز ذلك...

تتوفر مفاتيح مختلفة باستخدام أمر iostat. وفيما يلي أهم المفاتيح التي تستخدم جنبا إلى جنب مع iostat:

- d-: يعرض هذا المفتاح عدد الكيلوبايتات التي تنقل في الثانية لأقراص محددة، وعدد التحويلات في الثانية الواحدة، ومتوسط وقت الخدمة بالمللي ثانية. وفيما يلي استخدام الأمر iostat -d:

```
bash-3.2$iostat -d 5  5

sd0            sd2          sd3          sd4
Kps tps serv  Kps tps serv Kps tps serv Kps tps serv
1   0    53   57  5  145    19 1   89    0   0  14
140 14   16   0   0   0    785 31  21    0   0  0
8   1    15   0   0   0    814 36  18    0   0  0
11  1    82   0   0   26   818 36  19    0   0  0
0   0    0    1   0   22   856 37  20    0   0  0
```

- D-: يقوم رمز التبديل هذا بسرد القراءة في كل ثانية ، والكتابة في كل ثانية ، والنسبة المئوية لاستخدام القرص:

```
bash-3.2$ iostat -D 5 5

sd0            sd2          sd3           sd4
rps wps util rps wps util  rps wps util  rps wps util
0    0 0.3    4   0   6.2   1   1   1.8   0   0   0.0
0    0 0.0    0  35  90.6  237  0   97.8  0   0   0.0
0    0 0.0    0  34  84.7  218  0   98.2  0   0   0.0
0    0 0.0    0  34  88.3  230  0   98.2  0   0   0.0
```

```
0    2 4.4    0  37  91.3  225  0   97.7  0   0  0.0
```

- x-: سيقوم رمز التبديل هذا بالإبلاغ عن إحصائيات القرص الموسعة لكافة الأقراص:

```
bash-3.2$ iostat -x
                  extended device statistics
device      r/s    w/s   kr/s    kw/s wait actv  svc_t  %w  %b
fd0         0.0    0.0   0.0     0.0  0.0  0.0    0.0   0   0
sd0         0.0    0.0   0.4     0.4  0.0  0.0   49.5   0   0
sd2         0.0    0.0   0.0     0.0  0.0  0.0    0.0   0   0
sd3         0.0    4.6   0.0   257.6  0.0  0.1   26.4   0   12
sd4        69.3    3.6 996.9   180.5  0.0  7.6  102.4   0  100
nfs10       0.0    0.0   0.4     0.0  0.0  0.0   24.5   0   0
nfs14       0.0    0.0   0.0     0.0  0.0  0.0    6.3   0   0
nfs16       0.0    0.0   0.0     0.0  0.0  0.0    4.9   0   0
```

كيف تعمل...

يرسل الأمر iostat إحصاءات حول عمليات إدخال وإخراج القرص لإنتاج قياسات الإنتاجية، والاستخدام، وأطوال الطوابير، ومعدل المعاملة، ووقت الخدمة.

السطر الأول من الإخراج iostat يظهر كل شيء منذ إقلاع النظام، في حين يظهر كل سطر لاحق فقط الفاصل الزمني المحدد.

إذا لاحظنا الناتج السابق من iostat-d الأمر في قسم كيف ينجز ذلك...يمكننا أن نرى بوضوح أن محرك الأقراص sd3 مثقل بشكل كبير. فقيم أعمدة في kps (اختصار لكلمة الكيلو بايت المنقول في الثانية)، tps (اختصار لكلمة عدد التحويلات في الثانية)، و serv (اختصار لكلمة متوسط وقت الخدمة في ميلي ثانية) لمحرك الأقراص sd3 مرتفعة باستمرار على مدى الفاصل الزمني المحدد. وهذا يقودنا إلى استنتاج مفاده أن نقل المعلومات من sd3 إلى أي محرك أقراص آخر قد يكون فكرة جيدة إذا كانت هذه المعلومات تمثل إدخال/إخراج القرص على أساس ثابت. وهذا من شأنه أن يقلل من العبء على sd3.

أيضا، إذا لاحظت خرج الأمر iostat -D في قسم كيف ينجز ذلك...يمكنك استنتاج أن sd3 لديه نشاط قراءة عالي، والتي يشار إليها من قبل القيم العالية في العمود rps (اختصار لكلمة نسبة القراءة في الثانية) . وبالمثل، يمكننا أن نلاحظ أن محرك الأقراص sd2 يحدث فيه نشاط كتابة عالي، والتي يشار إليها من قبل القيم العالية في العمود wps (اختصار لكلمة نسبة الكتابة في الثانية). كل من محركات الأقراص، sd2 و sd3، في ذروة مستوى الاستخدام، والتي

يمكن أن ينظر إليها من نسبة عالية في العمود util (اختصار لكلمة الاستخدام) . تعتبر القيم العالية في عمود الاستخدام مؤشرا على مشاكل الإدخال / الإخراج، والتي يجب أن يحقق فيها من قبل مسؤول النظام.

وبالمثل، إذا كنا نلقي نظرة على الناتج السابق من الأمر iostat -x في قسم كيف ينجز ذلك...يمكننا أن نصل بسهولة إلى استنتاج مفاده أن محرك الأقراص sd4 يواجه مشاكل الإدخال / الإخراج كما رأينا من عمود b٪، مما يشير إلى النسبة المئوية من وقت القرص المشغول. لـ Sd4، ونسبة استخدام القرص هي مئة في المئة، والتي تحتاج إلى اهتمام فوري من مسؤول النظام.

مراقبة أداء النظام

في كثير من الأحيان، هناك حالات عندما تشكو تطبيقات المستخدمين من بطئ أداء لقاعدة البيانات، ونتيجة لذلك يحتاج مدير قاعدة البيانات إلى تحديد ما إذا كان هناك اختناقات في موارد النظام على خادم بوستجريسكل. تشغيل الأمر vmstat يمكن أن يساعدنا على تحديد الموقع بسرعة وتحديد أي اختناقات على الخادم.

الاستعداد للعمل.

نفذت الأوامر في هذا القسم على جهاز CentOS Linux.

كيف ينجز ذلك...

يستخدم الأمر vmstat للإبلاغ عن إحصاءات الأداء في الوقت الحقيقي حول العمليات والذاكرة والترحيل، وإدخال/ إخراج القرص واستهلاك وحدة المعالجة المركزية. وفيما يلي طريقة استخدام الأمر vmstat:

```
$ vmstat
procs --------memory--------  --swap-- --io-- -system- ---cpu---
r  b  swpd  free  buff cache   si  so   bi bo   in cs   us sy id wa
14 0  52340 25272 3068 1662704 0  0    63 76   9  31   15 1  84 0
```

في الخرج السابقة، يقسم السطر الأول الأعمدة على السطر الثاني إلى ست فئات مختلفة، والتي ستناقش على النحو التالي:

- الفئة الأولى هي العملية (procs) وتحتوي على الأعمدة التالية:

- r: يشير هذا العمود إلى العدد الإجمالي للعمليات التي تنتظر وقت التشغيل
- b: هذا العمود يبلغ إجمالي عدد العمليات في الخمول المتواصل

- الفئة الثانية هي الذاكرة memory وتحتوي على الأعمدة التالية:
  - swpd: يشير هذا العمود إلى إجمالي مقدار الذاكرة الافتراضية قيد الاستخدام
  - free: يعرض هذا العمود كمية الذاكرة الخاملة المتاحة للاستخدام
  - buff: يشير هذا العمود إلى مقدار الذاكرة المستخدمة في المخازن المؤقتة
  - cache: يشير هذا العمود إلى مقدار الذاكرة المستخدمة كذاكرة تخزين مؤقت للصفحة
- الفئة الثالثة هي التبادل swap، والتي تحتوي على الأعمدة التالية:
  - si: يعرض هذا العمود كمية الذاكرة التي أبدلت من داخل القرص
  - so: يشير هذا العمود إلى مقدار الذاكرة التي أبدلت من خارج القرص
- الفئة الرابعة هي الإدخال / الإخراج (io) ويحتوي على الأعمدة التالية:
  - bi: يشير هذا العمود إلى الكتل التي تقرأ من داخل جهاز كتلة
  - bo: هذا العمود تقارير كتل التي تكتب من الخارج إلى جهاز كتلة
- والفئة الخامسة هي النظام system الذي يتضمن الأعمدة التالية:
  - in: يعرض هذا العمود عدد الانقطاعات في الثانية الواحدة
  - cs: يعرض هذا العمود عدد مفاتيح السياق في الثانية.
- الفئة الأخيرة هي وحدة المعالجة المركزية (cpu) وتحتوي على الأعمدة التالية:
  - us: يعرض هذا العمود النسبة المئوية من الوقت الذي تدير فيه وحدة المعالجة المركزية رمز مستوى المستخدم
  - sy: يعرض هذا العمود النسبة المئوية للوقت الذي تدير فيه وحدة المعالجة المركزية رمز مستوى النظام
  - id: هذا العمود يقوم بتقارير النسبة المئوية من وقت خمول وحدة المعالجة المركزية
  - wa: يعرض هذا العمود مقدار الوقت المستغرق في انتظار الإدخال / الإخراج للإنهاء.

كيف تعمل...

وفيما يلي القواعد العامة المهمة المستخدمة أثناء تفسير إخراج الأمر vmstat.

- إذا كانت القيمة في عمود wa مرتفعة، فهذا مؤشر على أن نظام التخزين قد يكون مفرطا في التحميل وأنه يجب اتخاذ إجراء لمعالجة هذه المشكلة.

- إذا كانت القيمة في العمود b أكبر من الصفر بانتظام، فهذا مؤشر على أن النظام ليس لديه ما يكفي من طاقة المعالجة لخدمة الوظائف حارية التشغيل حاليا والمجدولة.

- إذا كانت القيم في الأعمدة so و si أكبر من الصفر عند مراقبتها لفترة من الزمن، فإنها تعد مؤشرا وأعراضا على وجود اختناق في الذاكرة.

فحص تحميل سجل وحدة المعالجة المركزية

في هذه الوصفة، سنقوم بإظهار كيفية استخدام الأمر sar في تركيبة مع مفاتيح متنوعة لتحليل سجل تحميل وحدة المعالجة المركزية في الوقت سابق.

الاستعداد للعمل

نفذت الأوامر المستخدمة في هذه الوصفة على حاسوب لينكس أوبونتو.

كيف ينجز ذلك...

يستخدم الأمر sar مع مفتاح -U لعرض إحصائيات وحدة المعالجة المركزية. وعند استخدامه بهذه الطريقة، فإن الأمر sar يقوم بالإبلاغ عن أنشطة هذا اليوم.

إذا كنا نتطلع إلى تحليل إحصاءات وحدة المعالجة المركزية منذ بعض الوقت، نحن بحاجة إلى استخدام التبديل -f بالتزامن مع المفتاح -U من الأمر sar. يتبع الخيار -f بالملفات التي يستخدمها sar للإبلاغ عن الإحصاءات لأيام مختلفة من الشهر. وعادة ما تكون هذه الملفات موجودة في الدليل /var/log/sa وعادة ما يكون لها اصطلاح تسمية sadd، حيث يمثل dd اليوم أو رقمي الشهر و تكون قيمته في نطاق من ٠١ إلى ٣١.

وفيما يلي استخدام الأمر sar لعرض إحصاءات وحدة المعالجة المركزية لليوم الثامن من الشهر:

```
$ sar -u -f /var/log/sa/sa08

03:50:10 PM CPU %user %nice %system  %iowait  %idle
04:00:10 PM all 0.42   0.00    0.24    0.00    96.41
04:10:10 PM all 0.22   0.00    1.96    0.00    95.53
```

```
04:20:10 PM all 0.22   0.00    1.22    0.01    99.55
04:30:10 PM all 0.22   0.00    0.24    2.11    99.54
04:40:10 PM all 0.24   0.00    0.23    0.00    92.54
Average:    all 0.19   0.00    0.19    0.07    99.55
```

كيف تعمل..

عموما القواعد المهمة هنا هي أنه إذا كانت قيمة idle% منخفضة، فإنه بمثابة مؤشر أنه إما ضعف طاقة وحدات المعالجة المركزية أو تحميل التطبيق مرتفع. وبالمثل، إذا رأينا القيم غير صفرية في عمود الوقت iowait، فإنه بمثابة تذكير بإمكانية اختناق نظام الإدخال / الإخراج الفرعي.

إذا لاحظنا الناتج السابق من الأمر sar، يمكننا أن نرى أن وقت idle% مرتفع، مما يدل بوضوح على أن وحدة المعالجة المركزية ربما غير مثقلة، ونحن لا نرى العديد من القيم غير الصفرية في العمود iowait%، الذي يخبرنا أنه ليس هناك الكثير من المنازعات على الإدخال / الإخراج في القرص.

هناك المزيد...

عند تثبيت حزمة sysstat، تجدول بعض مهام cron لإنشاء ملفات تستخدمها أداة sar للإبلاغ عن إحصائيات سجل الخادم. يمكننا مراقبة وظائف cron هذه بإلقاء نظرة على الملف /etc/cron.d/sysstat.

فحص سجل تحميل الذاكرة

في هذه الوصفة،سوف نرى كيفية تحليل تحميل الذاكرة ليوم سابق من الشهر.

الاستعداد للعمل

نفذت الأوامر المستخدمة في هذه الوصفة على حاسوب لينكس أوبونتو. قد يختلف أمر الإخراج في أنظمة التشغيل الأخرى القائمة على لينكس و يونكس.

كيف ينجز ذلك...

عندما يتعلق الأمر بتحليل إحصاءات الذاكرة، نحتاج إلى التحقق من إحصاءات الترحيل وإحصاءات التبادل.

يمكننا استخدام الأمر sar بالتزامن مع المفتاح B- للإبلاغ عن إحصاءات الترحيل جنبا إلى جنب مع المفتاح f- من أجل الإبلاغ عن إحصاءات لأيام مختلفة من الشهر. كما هو مذكور في الوصفة السابقة، الملفات التي يستخدمها الأمر sar للإبلاغ عن إحصائيات لأيام مختلفة من الشهر موجودة في الدليل /var/log/sa، ويصطلح عليها بـ sadd، حيث تمثل dd ترقيم الشهر، مع قيم تتراوح من ٠١ إلى ٣١.

على سبيل المثال، للإبلاغ عن إحصائيات الترحيل لليوم الخامس من الشهر، يمكننا استخدام الأمر sar كما يلي:

```
$ sar -B -f /var/log/sa/sa05

06:10:05 AM pgpgin/s  pgpgout/s  fault/s   majflt/s
06:20:05 AM 0.02        18.17      19.37     0.00
06:30:05 AM 4.49        26.68      76.15     0.05
06:40:05 AM 4512.43    419.24     380.14     0.65
06:50:06 AM 4850.03   1055.79    4364.73     0.51
07:00:06 PM 4172.68   1096.96    6650.51     0.16
```

وبالمثل، للإبلاغ عن إحصاءات التبادل لأيام مختلفة من الشهر، يمكننا استخدام الأمر sar بالتزامن مع المفتاح W- ومفتاح f-.

على سبيل المثال، للإبلاغ عن إحصاءات التبادل لليوم الخامس من الشهر، يمكننا استخدام الأمر sar على النحو التالي:

```
$ sar -W -f /var/log/sa/sa05

06:10:05 AM  pswpin/s pswpout/s
06:20:05 AM   0.00      0.00
06:30:05 AM   0.02      0.00
06:40:05 AM   1.15      1.45
06:50:06 AM   0.94      2.99
07:00:06 PM   0.67      6.95
```

كيف تعمل...

في الناتج السابق من الأمر sar -B -f /var/log/sa/sa05، يمكننا أن نرى بوضوح أنه في حوالي الساعة ٦.٤٠ صباحا، كان هناك زيادة كبيرة في الترحيل من القرص (pgpgin/s)، صفحات مقسمة من الخارج إلى القرص (pgpgout/s)، وأخطاء الصفحة في الثانية (fault/s).

وبالمثل، عندما يتم الإبلاغ عن إحصاءات التبادل مع الأمر sar -W -f /var/log/sa/ sa05، يمكننا أن نرى بوضوح أن التبادل بدأت في حوالي الساعة ٠٦.٤٠ صباحا، والتي يمكن أن ينظر إليها من القيم في العمود pswpin/s والعمود pswpout/s. إذا رأينا قيم عالية في العمود pswpin/s (الصفحات التي تم تبديلها إلى الذاكرة في الثانية) و pgpgout/s (الصفحات التي تم تبديلها إلى الخارج في الثانية)، فهذا يعني أن الذاكرة الحالية غير كافية وتحتاج إما إلى أن تزاد أو يغير حجمها على النحو الأمثل لبعض مكونات التطبيق.

مراقبة استخدام مساحة القرص

في هذه الوصفة، سنقوم بإظهار الأوامر التي تستخدم لمراقبة مساحة القرص.

الاستعداد للعمل

نفذت الأوامر في هذا القسم على خادم سولاريس.

كيف ينجز ذلك...

يمكننا استخدام الأمر df مع مفاتيح مختلفة لمراقبة مساحة القرص. لجعل الإخراج أكثر وضوحا، ونحن غالبا ما نستخدم المفتاح -h مع الأمر df:

```
bash-3.2$ df -h

Filesystem                                  size   used  avail capacity  Mounted on
                                            132G   80G    50G    62%      /
/devices                                     0K     0K     0K     0%      /devices
ctfs                                         0K     0K     0K     0%     /system/contract
proc                                         0K     0K     0K    0%      /proc
mnttab                                       0K     0K     0K    0%      /etc/mnttab
swap                                        418G   488K   418G     1%     /etc/svc/volatile
swap                                        418G    38M   418G     1%     /tmp
swap                                        418G   152K   418G     1%     /var/run
/dev/dsk/c20t60000970000192602156533030374242d0s0  236G   240M   234G     1%     /peterdata/cm_new

/dev/dsk/c20t60000970000192602156533032353441d0s0   30G    30M    29G     1%     /peterdata/native
```

```
/dev/dsk/c20t60000970000192602156533033313441d0s0    236G    60G   174G    26%
/peterdata/db_new
/dev/dsk/c20t60000970000195701036533032454646d0s0     30G   6.9G    22G    24%    /peterdata/native

/dev/dsk/c20t60000970000195701036533032444137d0s0    236G   224G    12G    95%    /peterdata/db

/dev/dsk/c20t60000970000192602156533032333232d0s2    709G   316G   386G    45%    /peterdata/cm

usmtnnas4106-epmnfs.emrsn.org:/peterb1ap_2156    276G   239G    36G    87%    /peterb1ap
usmtnnas4106-epmnfs.emrsn.org:/peterdata_data_2156     98G    53G    45G    54%    /peterdata/data
usmtnnas4106-epmnfs.emrsn.org:/peterdata_uc4appmgr    9.8G   3.6G   6.3G    37%    /peterdata/uc4/
```

كيف ينجز ذلك...

إذا لاحظنا الناتج السابق، يمكننا أن نرى أن النقطة الربط لـ /peterdata/db تقترب من كامل قدرتها (وصلت إلى قدرة ٩٥٪) وبقي فقط ١٢ غيغابايت من مساحة القرص الحرة متاحة على الجهاز. هذا مؤشر على أن المسؤول يحتاج إلى إما تنظيف بعض الملفات القديمة من على قرص المربوط لتوفير المزيد من المساحة الحرة، أو تخصيص مساحة إضافية إلى نقطة الربط المعطاه قبل أن تصل إلى كامل طاقتها.

مراقبة حالة الشبكة

في هذه الوصفة، سنقوم بعرض كيفية مراقبة حالة واجهات الشبكة.

الاستعداد للعمل

نفذت الأوامر المستخدمة في هذه الوصفة على حاسوب سينتوس لينكس.

قد يختلف أمر الإخراج في أنظمة التشغيل الأخرى القائمة على لينوكس أو يونيكس.

كيف ينجز ذلك...

سنقوم باستخدام الأمر netstat مع المفتاح i- لعرض حالة واجهات الشبكة التي تم تضبيطها على النظام. في ما يلي لقطة شاشة توضح استخدام الأمر netstat:

```
bash-3.2$ netstat -i
Name  Mtu  Net/Dest      Address       Ipkts  Ierrs Opkts  Oerrs Collis Queue
lo0   8232 loopback      localhost     187358785 0     187358785 0     0      0
nxge909000 1500 usmtnz-sinfsi17.emrsn.com usmtnz-sinfsi17.emrsn.com 619320037 0     915300941 0     0      0
nxge909002 0    default       0.0.0.0       146274 0     0      0     0      0
nxge2761000 1500 0.0.0.0       0.0.0.0       619320058 0     915301069 0     0      0
nxge2761002 0    default       0.0.0.0       146274 0     0      0     0      0
sppp0 1500 192.168.224.1 192.168.224.3  13421  0     13489  0     0      0
```

كيف تعمل..

في الخرج السابق للأمر netstat -i، يمكننا تحديد عدد الحزم التي يقوم النظام بإرسالها واستقبالها على كل واجهة شبكة. يحدد العمود Ipkts عدد حزم الإدخال ويحدد العمود Okpts عدد حزم الإخراج. إذا ظل عدد حزم الإدخال ثابتا على مدى فترة من الزمن، فهذا يعني أن الجهاز لا يتلقى حزم الشبكة على الإطلاق، وتشير النتيجة إلى فشل من جهة العتاد في واجهة الشبكة. وإذا ظل عدد حزم الخرج ثابتا على مدى فترة من الزمن ، فقد يعني ذلك احتمال حدوث مشاكل بسبب إدخال عنوان غير صحيح في قاعدة البيانات الخاصة بالمضيف أو قاعدة البيانات ethers.

٦- مراقبة نشاط قاعدة البيانات والتحقيق في مشاكل الأداء

في هذا الفصل، سنغطي الوصفات التالية:

- فحص الجلسات النشطة
- التعرف على طلبات البحث التي يجريها المستخدمون حاليا
- الحصول على خطة التنفيذ لبيان معين
- تسجيل البيانات البطيئة
- جمع الإحصاءات
- مراقبة حمل قاعدة البيانات
- العثور على جلسات حظر
- إحصاءات النفاذ إلى الجدول
- إيجاد الفهارس غير المستخدمة
- فرض استعلام لاستخدام الفهرس
- تحديد استخدام القرص

مقدمة

تعتبر مراقبة قواعد البيانات وحل مشاكل الأداء إحدى المهام الرئيسية لمسؤول قاعدة البيانات. فضمان قاعدة بيانات صحية تمتاز بالأداء الأمثل هو ما يوظف مدير قاعدة البيانات لأجله. يجب أن تراقب قاعدة البيانات بانتظام بطريقة استباقية لحل أي مشاكل معروفة قبل أن تصل إلى حالة حرجة وتؤدي إلى الأداء السيئ.

استكشاف مشاكل الأداء هي طريقة عمل تعتمد على رد الفعل لأنه فقط بعد أن يبلغ عن مشكلة يبدأ مدير قاعدة البيانات باستكشاف الأخطاء وإصلاحها. إذا عينت التنبيهات الاستباقية في الوقت المناسب، يمكن أن يضمن المدير أن قاعدة البيانات في حالة صحية، وهذا يمكن أن يؤدي إلى انخفاض في مشاكل الأداء.

لقد استخدمنا عينة قاعدة البيانات dvdrental لكل الشفرات في هذا الفصل. عينة dvdrental متوفرة في حزمة الشفرات. تفاصيل حول تركيب وعمل قاعدة البيانات dvdrental متوفرة أيضا في حزمة الشفرات.

التحقق من الجلسات النشطة

في هذه الوصفة، سوف نتعلم كيفية التحقق من وجود جلسات نشطة في قاعدة بيانات.

الاستعداد للعمل

سنقوم بالاستعلام عن جدول pg_stat_activity للتحقق من وجود جلسات نشطة في قاعدة بيانات. الاستعلام المستخدم في هذه الوصفة يعمل في بوستجريسكل الإصدار ٩.٢ فما فوق.

كيف ينجز ذلك...

يمكننا استخدام استعلام SQL التالي للعثور على الجلسات النشطة في قاعدة البيانات hrdb:

```
SELECT pid , usename, application_name, client_addr, client_hostname, query,
state FROM pg_stat_activity
WHERE datname='dvdrental';
```

كيف تعمل..

نستخدم الاستعلام السابق للعثور على كافة اتصالات العميل التي أجريت على قاعدة البيانات hrdb.

فيما يلي شرح للأعمدة في الجدول pg_stat_activity للعثور على معلومات حول الجلسات النشطة في قاعدة بيانات hrdb :

- عمود pid: تشير القيمة في هذا العمود إلى معرف العملية للمستخدم المتصل حاليا إلى قاعدة البيانات، وهي قاعدة البيانات hrdb في حالتنا.

- عمود datname: تشير القيمة في هذا العمود إلى اسم قاعدة البيانات التي يتصل بها المستخدم حاليا.
- عمود application_name: توفر القيمة في هذا العمود اسم التطبيق الذي يستخدمه المستخدم المتصل حاليا بقاعدة البيانات.
- عمود client_addr : القيمة في هذا العمود تعطي عنوان IP الخاص بالمستخدم المتصل حاليا بقاعدة البيانات.
- عمود client_hostname: تعطي القيمة في هذا العمود اسم المضيف للعميل المتصل.
- عمود query: توفر القيمة الموجودة في هذا العمود النص الكامل لاستعلام SQL الذي ينفذه العميل.

يتضمن الناتج السابق أيضا عمود state، الذي يشير إلى حالة عمود pid للمستخدم المتصل حاليا. يمكن أن يكون لعمود state القيم المحتملة التالية:

- active: تشير هذه القيمة إلى أن جلسة المستخدم تقوم حاليا بتنفيذ الاستعلام في الخلفية.
- idle: تشير هذه القيمة إلى أن الواجهة الخلفية تنتظر أمر جديد للعميل.
- idle in transaction: تشير هذه القيمة إلى أن عملية الواجهة الخلفية متضمنة حاليا في معاملة ولكنها لا تنفذ استعلام.
- fastpath function call: تشير هذه القيمة إلى أن عملية الواجهة الخلفية تقوم بتنفيذ وظيفة المسار السريع.
- disabled: يبلغ عن هذه القيمة إذا عطلت قيمة معلمة التضبيط track_activities للوصلة الخلفية المتصلة حاليا. إذا عطلت قيمة عمود الحالة، فهذا يعني أن ميزة جمع المعلومات لا تعمل على الأمر الذي ينفذ حاليا لكل جلسة.

هناك المزيد...

إذا كنت تستخدم إصدار بوستجريسكل قبل ٩.٢، يمكنك استخدام الاستعلام التالي للعثور على جلسات عمل نشطة في قاعدة بيانات بوستجريسكل:

```
SELECT datname , procpid, usename,application_name,client_addr,
client_hostname,current_query FROM pg_stat_activity;
```

التعرف على طلبات البحث التي يجريها المستخدمون حاليا

في هذه الوصفة، سنقوم بعرض الاستعلامات الأحدث أو التي تنفذ حاليا من قبل المستخدمين في قاعدة بيانات بوستجريسكل.

الاستعداد للعمل

قبل معرفة الاستعلامات التي يصدرها المستخدمون ضد قاعدة البيانات، أول شيء نحتاج إلى القيام به هو تفعيل معلمة التضبيط track_activities في ملف التضبيط postgresql.conf، كما يلي:

```
track_activities = on
```

بعد تفعيل هذه المعلمة، سنحتاج إلى إعادة تحميل ملف التضبيط للتأكد من أن التغييرات التي أدخلت سارية المفعول:

```
pg_ctl -D $PGDATA reload
```

كيف ينجز ذلك...

يمكننا استخدام الاستعلام التالي لعرض نص الاستعلام الذي ينفذ بواسطة المستخدم المتصل حاليا بقاعدة البيانات:

```
SELECT datname, pid, usename, query_start, state, query
FROM pg_stat_activity
```

يمكن استخدام هذا الاستعلام أيضا في محرر SQL من أداة pgAdmin .

كيف تعمل...

ستقوم بوستجريسكل بجمع بيانات حول كل الاستعلامات قيد التشغيل كلما فعلت معلمة تضبيط track_activities.

يمكننا مشاهدة الاستعلام الأخير الذي نفذ بواسطة مستخدم في أي قاعدة بيانات بوستجريسكل بالإشارة إلى عبارة SQL المستردة من عمود الاستعلام في جدول pg_stat_activity. يشير العمود query_start إلى الوقت على الخادم عندما نفذ العميل الاستعلام.

الحصول على خطة التنفيذ للتعليمة

في هذه الوصفة سنذهب لرؤية كيفية الحصول على خطة التنفيذ لتعليمة SQL.

الاستعداد للعمل

يتم استخدام الأمر EXPLAIN للحصول على خطة التنفيذ لتعليمة SQL.

كيف ينجز ذلك...

يحتوي كل استعلام ينفذ في بوستجريسكل على خطة تنفيذ. يمكن تشغيل الأمر EXPLAIN في أي من الأوضاع الثلاثة المعطاة:

- الوضع العام: في هذا الوضع، نحن بحاجة فقط إلى تحديد الأمر EXPLAIN متبوعا بعبارة SQL. سيقوم مخطط بوستجريسكل بعرض خطة التنفيذ التي أنشأت لتعليمة SQL المحددة. ستعرض خطة التنفيذ طريقة الفحص المستخدمة للوصول إلى الجدول المشار إليه في الاستعلام. تفاصيل أخرى متضمنة يمكن أن تكون تكلفة التنفيذ المقدرة لتعليمة SQL، وهو تقدير المخطط لكم من الوقت المستغرق لتنفيذ تعليمة SQL. يمكن استدعاء الأمر EXPLAIN كما يلي:

```
dvdrental=# EXPLAIN  select * from payment where amount > 4.99;
                        QUERY PLAN
-------------------------------------------------------------
 Seq Scan on payment  (cost=0.00..290.45 rows=3616 width=26)
   Filter: (amount > 4.99)
(2 rows)
```

- وضع التحليل: يمكن أيضا تنفيذ تعليمة SQL في وضع التحليل. يوفر هذا إحصائيات وقت التشغيل الفعلي مثل الوقت الإجمالي الذي يستغرقه تنفيذ الاستعلام والعدد الفعلي للصفوف التي أرجعت. وبمساعدة من هذا الخيار يمكننا تحديد ما إذا كانت تقديرات مخطط بوستجريسكل قريبة من الأرقام الفعلية أم لا. يمكننا تشغيل وضع EXPLAIN ANALYZE على النحو التالي:

```
dvdrental=# EXPLAIN ANALYZE select * from payment where amount > 4.99;
                        QUERY PLAN
------------------------------------------------------------------
 Seq Scan on payment  (cost=0.00..290.45 rows=3616 width=26) (actual time=0.024.
```

```
.7.117 rows=3618 loops=1)
   Filter: (amount > 4.99)
   Rows Removed by Filter: 10978
 Total runtime: 7.457 ms
(4 rows)
```

تنزيل كود المثال

يمكنك تنزيل ملفات كود المثال لجميع كتب باكت التي اشتريتها من حسابك على

http://www.packtpub.com

إذا اشتريت هذا الكتاب في مكان آخر، فيمكنك زيارة

http://www.packtpub.com/support

والتسجيل للحصول على ملفات البريد الإلكتروني التي يتم إرسالها إليك مباشرة.

- الوضع المطنب: الاستفادة من تشغيل الأمر EXPLAIN في وضع مطنب هو أن إخراج خطة EXPLAIN سوف يعرض أيضا الأعمدة التي مررت بواسطة الاستعلام. يمكن أن تكون هذه المعلومات قيمة عندما يكون الاستعلام الأساسي معقدا.

يمكننا تشغيل الأمر EXPLAIN في وضع مطنب كما يلي:

```
dvdrental=# EXPLAIN VERBOSE select * from payment where amount > 4.99;
                            QUERY PLAN
--------------------------------------------------------------------------
 Seq Scan on public.payment  (cost=0.00..290.45 rows=3616 width=26)
   Output: payment_id, customer_id, staff_id, rental_id, amount, payment_date
   Filter: (payment.amount > 4.99)
(3 rows)
```

كيف تعمل...

ينظم إخراج الأمر EXPLAIN في سلسلة من عقد الخطة. تحلل بنهج من أسفل إلى أعلى. في الأسفل، هناك العقد التي تنظر في الجداول، قم بمسحها، أو انظر إلى الأشياء من خلال فهرس. كل سطر في إخراج أمر EXPLAIN هو عقدة خطة.

هناك العديد من القياسات الرقمية التي ترتبط مع العقدة. على سبيل المثال، إذا نظرنا إلى إخراج الأمر EXPLAIN ANALYZE، يمكننا أن نرى التفاصيل التالية:

- Seq Scan: أول شيء نلاحظه هو أن الخطة لديها عقدة واحدة، وهي عقدة المسح المتسلسل.
- cost=0.00..290.45: التكلفة الأولى هي تكلفة بدء هذه العقدة. تحدد القيمة هنا مقدار العمل المقدر أن يتم قبل أن تنتج العقدة الصف الأول من الإخراج. هنا، القيمة صفر لأن عقدة "Seq Scan" تنتج الصفوف على الفور. التكلفة المقدرة الثانية هي تكلفة تشغيل العقدة بأكملها حتى تكتمل.
- rows=3616: عدد الصفوف للإخراج إذا تم تشغيل العقدة إلى الاكتمال.
- width=26: توفر هذه القيمة تقديرا لمتوسط عدد وحدات البايت لكل إخراج صف للعقد.

وتتعلق النقاط التي ناقشناها للتو بالقيم المقدرة. الأرقام الفعلية تخبر عن تفاصيل وقت الاستجابة للاستعلام. ويتألف الوقت الفعلي من التكلفة الفعلية لبدء التشغيل وتكلفة تشغيل العقدة بأكملها. يعرض عمود الصفوف العدد الفعلي للصفوف التي يرجعها الاستعلام.

إذا كان الفرق بين الصفوف المقدرة والصفوف الفعلية ضخم، فهذا مؤشر على أن محسن الاستعلام قد اتخذ قرارا سيئا بناء على خطة التنفيذ الحالية.

لمزيد من المعلومات التفصيلية، يرجى الرجوع إلى

http://www.postgresql.org/docs/9.2/static/sql-explain.html

الذي يصف خطط التنفيذ في بوستجريسكل.

تسجيل التعليمات البطيئة

في هذه الوصفة، نذهب لتغطية كيفية تسجيل تعليمات بطيئة في خادم بوستجريسكل.

الاستعداد للعمل

سوف نحتاج إلى إجراء تغييرات على بعض معلمات التضبيط في الملف postgresql.conf و التي تقوم بتفعيل التسجيل ثم قم بإعادة تشغيل خادم بوستجريسكل من أجل التأكد من أن التغييرات التي أجريت على معلمات التضبيط هذه سارية المفعول.

كيف ينجز ذلك...

هنا هو تسلسل الخطوات التي يجب اتباعها من أجل تسجيل تعليمات SQL المشتغلة ببطء:

١. يجب أن تعين المعلمات التالية في الملف postgresql.conf:

```
logging_collector = on
log_directory = 'pg_log'
log_min_duration_statement = 100
```

٣. بمجرد تعيين هذه المعلمات في الملف postgresql.conf، سوف نحتاج إلى إعادة تشغيل خادم بوستجريسكل كما يلي:

```
pg_ctl -D $PGDATA restart
```

كيف تعمل...

وفيما يلي شرح لتسلسل الخطوات المتخذة في القسم السابق:

- إعداد log_min_duration_statement إلى ١٠٠، كما رأينا في القسم السابق،

يعني أنه ستسجل كل تعليمات SQL التي تنفذ لمدة ١٠٠ ميلي ثانية أو أكثر في خادم بوستجريسكل. هذه معلمة مفيدة يحب تفعيلها لأنه يمكن أن تساعد في تتبع الاستعلامات غير المحسنة في تطبيقات العميل.

- إعداد المعلمة logging_collector تمكن جامع التسجيل، عملية خلفية وظيفتها لالتقاط رسائل السجل المرسلة إلى stderr وإعادة توجيهها إلى ملفات السجل. يعد تعيين هذه المعلمة مفيدا لأن رسائل السجل التي التقطت بهذه الطريقة قد تحتوي على معلومات أكثر من syslog.
- تعيين المعلمة log_directory سيعين الدليل الذي سينشأ فيه ملفات السجل.

جمع الإحصاءات

في هذه الوصفة، سنغطي المعلمات التي تحتاج إلى تفعيل من أجل جمع الإحصاءات.

الاستعداد للعمل

يأتي خادم بوستجريسكل مع مجموعة من وظائف النفاذ إلى الإحصاءات معرفة مسبقا ومجموعة من عروض الإحصاءات المعرفة مسبقا. تستخدم هذه العروض وظائف الإحصائيات المحددة لتجميعها

في بوستجريسكل. بشكل افتراضي، يجمع عدد صغير فقط من الإحصاءات. في القسم التالي، سوف نقوم بتغطية معلمات التضبيط التي تتحكم في جمع الإحصاءات.

كيف ينجز ذلك...

هذا هو تسلسل الخطوات التي يجب اتباعها من أجل تفعيل جمع الإحصاءات في بوستجريسكل:

١. يجب أن تعين معلمات التضبيط التالية في الملف postgresql.conf:

```
track_activities = on
track_counts = on
track_functions = all
track_io_timing = on
```

٢. بعد تعيين معلمات التضبيط هذه، سنحتاج إلى إعادة تحميل ملف التضبيط لضمان إدخال تغييرات المعلمات:

```
pg_ctl -D $PGDATA reload
```

كيف تعمل...

وفيما يلي شرح للخطوات المتخذة في القسم السابق.

- وضع track_activities تمكن مراقبة الأمر قيد التنفيذ حاليا من قبل عملية الخادم، جنبا إلى جنب مع الوقت الذي بدأ فيه تنفيذ الأمر.
- تعيين معلمة التضبيط track_counts تمكن جمع الإحصاءات عن نشاط قاعدة البيانات، والذي يتضمن جمع الإحصاءات للحصول على الجدول ونفاذ الفهرس.
- تحديد قيمة معلمة التضبيط track_functions للجميع تمكن من تتبع وظائف محددة من قبل المستخدم، والتي تشمل تتبع وظائف اللغة الإجرائية جنبا إلى جنب مع وظائف لغة SQL و لغة C.
- ضبط معلمة التضبيط track_io_timing يمكن توقيت مكالمات الإدخال/ الإخراج لقاعدة البيانات. قد يؤدي تفعيل هذه المعلمة إلى بعض الكلفة في الأداء لأنه عندما تفعل هذه المعلمة، تقوم بوستجريسكل بالتحقق بشكل متكرر من نظام التشغيل عن وقت الحالي. يسجل توقيت الإدخال/ الإخراج في عرض pg_stat_database. القطع الرئيسية من المعلومات التي تلتقط هي عدد كتل القرص والوقت الذي يقضيه على القراءة وكتابة كتل قاعدة البيانات لقاعدة بيانات بوستجريسكل معينة.

لمزيد من التفاصيل حول هذا الموضوع، يرجى الرجوع إلى

http://www.postgresql.org/docs/9.3/static/monitoring-stats.html

و

http://www.postgresql.org/docs/9.3/static/runtime-configstatistics.html#GUC-TRACK-ACTIVITIES.

مراقبة حمل قاعدة البيانات

في هذه الوصفة، سنقوم باستخدام الاستعلامات التي يمكن استخدامها لمراقبة تحميل قاعدة البيانات.

الاستعداد للعمل

يمكننا استخدام عرض pg_stat_database لمراقبة حمل قاعدة البيانات الحالية.

كيف ينجز ذلك...

ومن أجل تحديد الحمل الحالي لقاعدة البيانات، سنحتاج إلى معرفة ما يلي:

- معلومات مثل عدد اتصالات قاعدة البيانات النشطة
- عدد الالتزامات وعمليات التراجع الصادرة
- مجموعة كتل قراءة ونسبة التخزين المؤقتة لقاعدة بيانات معينة

يمكننا استخدام الاستعلام التالي لتحديد حمل قاعدة البيانات الموجودة لقاعدة بيانات dvdrental:

```
dvdrental=# SELECT numbackends as CONN, xact_commit as TX_COMM,
xact_rollback as
TX_RLBCK, blks_read + blks_hit as READ_TOTAL,
blks_hit * 100 / (blks_read + blks_hit)
as BUFFER FROM pg_stat_database WHERE datname = 'dvdrental';

 conn | tx_comm | tx_rlbck | read_total | buffer
------+---------+----------+------------+--------
    9 |      45 |        1 |       1456 |     99
(1 row)
```

كيف تعمل...

تعرض الأعمدة التالية بواسطة الاستعلام السابق:

- numbackends: يمثل هذا العمود إجمالي عدد الاتصالات النشطة.
- xact_commit: يمثل هذا العمود العدد الإجمالي للالتزامات.
- xact_rollback: يمثل هذا العمود إجمالي عدد عمليات التراجع
- blks_read: يمثل هذا العمود إجمالي الكتل التي قرأت.
- blks_hit: يمثل هذا العمود العدد الإجمالي للنتائج المخزنة مؤقتا.

هنا هو تسلسل الخطوات المطلوبة من أجل تحديد حمل قاعدة البيانات الحالية:

١. أولا، نحن بحاجة إلى إعادة تعيين الإحصاءات باستخدام وظيفة pg_stat_reset() ، مثل هذا:

```
dvdrental=# SELECT pg_stat_reset();
pg_stat_reset
---------------
(1 row)
```

٢ - وتتمثل الخطوة التالية في الانتظار لفترة من الوقت لضمان جمع إحصاءات كافية.

٣. الخطوة الأخيرة هي استدعاء الاستعلام الإحصائي على عرض pg_stat_database، كما هو موضح في القسم السابق.

العثور على جلسات حظر

في هذه الوصفة، سنرى الاستعلامات التي يمكن أن تساعدنا في معرفة جلسات المستخدمين التي حُضِرَث ومن حضرها.

الاستعداد للعمل

لتشغيل هذه الاستعلامات، ستحتاج إلى استخدام حساب المستخدم ذي الصلاحيات المطلقة superuser.

كيف ينجز ذلك...

يمكننا استخدام الاستعلام التالي للعثور على معلومات بشأن الحضر والجلسات المحظورة:

```
SELECT bl.pid AS blocked_pid,
  a.usename AS blocked_user,
  ka.query AS blocking_statement,
  now() - ka.query_start AS blocking_duration,
  kl.pid AS blocking_pid,
  ka.usename AS blocking_user,
  a.query AS blocked_statement,
  now() - a.query_start AS blocked_duration
FROM  pg_catalog.pg_locks bl
JOIN pg_catalog.pg_stat_activity a  ON a.pid = bl.pid
JOIN pg_catalog.pg_locks kl ON kl.transactionid = bl.transactionid AND kl.pid !
= bl.pid
JOIN pg_catalog.pg_stat_activity ka ON ka.pid = kl.pid
WHERE NOT bl.granted;
```

يعمل الاستعلام المذكور أعلاه في الإصدار ٩.٢ من بوستجريسكل والإصدارات اللاحقة.

كيف تعمل...

الاستعلام في المقطع السابق يجد معرف العملية واسم المستخدم والاستعلامات التي تشغلها الجلسات المحظورة، هنا نحن نستخدم شرط JOIN على العمود pid من جداول pg_locks و pg_stat_activity مرتين: مرة لجلسات الحظر ثم للجلسات المحظورة. نقوم أيضا بالانضمام إلى الجدول pg_lock نفسه، في العمود transactionid، وشرط التصفية هنا هو أن العمود (معرف العملية) pid يجب أن يكون غير مكرر(فريد) عند انضمامه إلى نفس الجدول pg_locks.

إذا كنت تستخدم إصدار بوستجريسكل أقدم من بوستجريسكل ٩.٢، فيمكنك استخدام هذا الاستعلام لتحديد جلسات الحظر:

```
SELECT bl.pid AS blocked_pid,
  a.usename AS blocked_user,
  ka.current_query AS blocking_statement,
  now() - ka.query_start AS blocking_duration,
  kl.pid AS blocking_pid,
  ka.usename AS blocking_user,
  a.current_query AS blocked_statement,
  now() - a.query_start AS blocked_duration
FROM  pg_catalog.pg_locks bl
JOIN pg_catalog.pg_stat_activity a  ON a.procpid = bl.pid
```

```
JOIN pg_catalog.pg_locks kl ON kl.transactionid = bl.transactionid AND kl.pid !
= bl.pid
JOIN pg_catalog.pg_stat_activity ka ON ka.procpid = kl.pid
WHERE NOT bl.granted;
```

إحصائيات النفاذ إلى الجدول

في هذه الوصفة، سنرى تفاصيل كيفية النفاذ إلى الجداول.

الاستعداد للعمل

تتوفر القيم الإحصائية حول جداول المستخدم في عرض pg_stat_user_tables. يمكن استخدام هذا الجدول للحصول على تفاصيل مثل العدد المقدر للصفوف الحية والميتة والطوابع الزمنية للوقت الذي كان فيه آخر تنظيف أو تنظيف تلقائي. وبالمثل، يمكننا استخدام pg_stat_user_tables للعثور على تفاصيل حول النفاذ إلى الجدول.

كيف ينجز ذلك...

يمكننا استخدام الاستعلام التالي لتحديد ما إذا كان يوصل إلى الجداول بواسطة فحوصات متتابعة أو فحوصات فهرسية:

```
dvdrental=# SELECT schemaname,relname,seq_scan,idx_scan,cast(idx_scan AS
numeric) / (idx_scan + seq_scan) AS idx_scan_pct FROM pg_stat_user_tables WHERE
(idx_scan + seq_scan)>0 ORDER BY idx_scan_pct;
 schemaname |    relname     | seq_scan | idx_scan |      idx_scan_pct
------------+----------------+----------+----------+--------------------
 public     | category       |        2 |        0 | 0.00000000000000000000
 public     | actor          |        3 |        0 | 0.00000000000000000000
 public     | customer       |        7 |        0 | 0.00000000000000000000
 public     | country        |        2 |        0 | 0.00000000000000000000
 public     | film_category  |        3 |        0 | 0.00000000000000000000
 public     | payment        |        7 |        0 | 0.00000000000000000000
 public     | inventory      |        4 |        0 | 0.00000000000000000000
 public     | language       |        2 |        0 | 0.00000000000000000000
 public     | store          |        4 |        0 | 0.00000000000000000000
 public     | film_actor     |        4 |        0 | 0.00000000000000000000
 public     | city           |        4 |        0 | 0.00000000000000000000
 public     | rental         |        7 |        0 | 0.00000000000000000000
```

```
 public       | staff           |        5 |          0 | 0.00000000000000000000
 public       | film            |        8 |          0 | 0.00000000000000000000
 public       | address         |        4 |          4 | 0.50000000000000000000
(15 rows)
```

كيف تعمل...

في الخرج السابق، يمكننا أن نرى أنه بالنسبة لغالبية الجداول بدءا من جدول category حتى جدول film، ينفذ إليها بواسطة عمليات فحوصات متتابعة لأن كافة البيانات تناسب صفحة بيانات واحدة. بالنسبة إلى عنوان الجدول، يمكننا أن نرى أنه بالنسبة لبعض الاستعلامات، يوصل إليها بواسطة فحوصات متتابعة، وبالنسبة لبعض العبارات، يستخدم بوستجريسكل الفهارس للبحث عن القيم.

جانب آخر مثير للاهتمام هو معرفة عدد الصفوف التي عولجت بواسطة هذه الفحصوصات.

يمكننا استخدام الاستعلام التالي للحصول على هذه المعلومات:

```
dvdrental=# SELECT relname,seq_tup_read,idx_tup_fetch,cast(idx_tup_fetch AS
numeric) / (idx_tup_fetch + seq_tup_read) AS idx_tup_pct FROM
pg_stat_user_tables WHERE (idx_tup_fetch + seq_tup_read)> 0 ORDER BY
idx_tup_pct;
    relname     | seq_tup_read | idx_tup_fetch |          idx_tup_pct
---------------+--------------+---------------+----------------------------
 category      |           32 |             0 |      0.00000000000000000000
 actor         |          600 |             0 |      0.00000000000000000000
 customer      |         4193 |             0 |      0.00000000000000000000
 country       |          218 |             0 |      0.00000000000000000000
 film_category |         3000 |             0 |      0.00000000000000000000
 payment       |       102172 |             0 | 0.000000000000000000000000
 inventory     |        18324 |             0 | 0.000000000000000000000000
 language      |           12 |             0 |      0.00000000000000000000
 store         |            8 |             0 |      0.00000000000000000000
 film_actor    |        21848 |             0 | 0.000000000000000000000000
 city          |         2400 |             0 |      0.00000000000000000000
 rental        |       112308 |             0 | 0.000000000000000000000000
 staff         |           10 |             0 |      0.00000000000000000000
 film          |         8000 |             0 |      0.00000000000000000000
 address       |         2412 |             4 |      0.00165562913907284768
(15 rows)
```

في المخرجات السابقة، يمكننا أن نرى أنه بالنسبة لكافة الجداول، عولجت معظم الصفوف بواسطة الفحص التتابعي. ووصل فقط إلى أربعة صفوف في جدول العناوين باستخدام بحث الفهرس.

العثور على فهارس غير مستخدمة

يصبح من الضروري التحقق من الفهارس غير المستخدمة لأن الفهارس في نهاية المطاف تستهلك جزءا كبيرا من مساحة القرص، وإذا لم تراقب عن كثب، فإنها يمكن أن تستهلك دورات وحدة المعالجة المركزية غير ضرورية، أكثر من ذلك فإنها تصبح مجزأة .

الاستعداد للعمل

من أجل أن تكون قادرا على العثور على الفهارس غير المستخدمة في بوستجريسكل، نحن بحاجة إلى التأكد من تفعيل معلمات التضبيط track_activities و track_counts في الملف postgresql.conf. فقط عندما تجمع الإحصاءات سنكون قادرين على تحديد الفهارس غير المستخدمة.

كيف ينجز ذلك...

يمكننا استخدام الاستعلام التالي لتحديد الفهارس غير المستخدمة في بوستجريسكل:

```
SELECT
    relid::regclass AS table,
    indexrelid::regclass AS index,
    pg_size_pretty(pg_relation_size(indexrelid::regclass)) AS index_size,
    idx_tup_read,
    idx_tup_fetch,
    idx_scan
FROM pg_stat_user_indexes
JOIN pg_index USING (indexrelid)
WHERE idx_scan = 0
AND indisunique IS FALSE;
   table    |         index         | index_size | idx_tup_read | idx_tup_fetch | idx_scan
------------+-----------------------+------------+--------------+---------------+----------
 film       | film_fulltext_idx     | 88 kB      |            0 |             0 |        0
 actor      | idx_actor_last_name   | 16 kB      |            0 |             0 |        0
 customer   | idx_fk_address_id     | 32 kB      |            0 |             0 |        0
 address    | idx_fk_city_id        | 32 kB      |            0 |             0 |        0
 city       | idx_fk_country_id     | 32 kB      |            0 |             0 |        0
```

```
 payment    | idx_fk_customer_id   | 336 kB     |           0 |               0 |          0
 film_actor | idx_fk_film_id       | 136 kB     |           0 |               0 |          0
 rental     | idx_fk_inventory_id  | 368 kB     |           0 |               0 |          0
 film       | idx_fk_language_id   | 40 kB      |           0 |               0 |          0
 payment    | idx_fk_rental_id     | 336 kB     |           0 |               0 |          0
 payment    | idx_fk_staff_id      | 336 kB     |           0 |               0 |          0
 customer   | idx_fk_store_id      | 32 kB      |           0 |               0 |          0
 customer   | idx_last_name        | 32 kB      |           0 |               0 |          0
 inventory  | idx_store_id_film_id | 120 kB     |           0 |               0 |          0
 film       | idx_title            | 56 kB      |           0 |               0 |          0
(15 rows)
```

كيف تعمل...

إذا قمنا بإلقاء نظرة على المخرجات السابقة، يمكننا أن نستنتج أنه أينما كان إدخال idx_scan يساوي صفر، يعني بوضوح أنه إما لم يستخدم الفهرس المعطى أبدا أو على الأرجح لم يستخدم منذ وقت تشغيل الوظيفة(pg_stat_reset) ، والتي تقوم بإعادة تعيين كافة عدادات الإحصاءات لقاعدة البيانات الحالية إلى الصفر. في القسم السابق، نقوم بعمل ارتباط بين جداول pg_stat_user_indexes و pg_index في العمود indexrelid.

في خرج الاستعلام السابق، تشير الأعمدة idx_tup_read و idx_tup_fetch و idx_scan إلى استخدام الفهرس:

- يشير العمود idx_tup_read إلى عدد الصفوف التي قرأت باستخدام الفهرس
- يشير العمود idx_tup_fetch إلى عدد الصفوف التي جلبت باستخدام الفهرس
- يشير العمود idx_scan إلى عدد مرات استخدام الفهرس بواسطة مخطط الاستعلام

هناك المزيد...

كما هو الحال مع الفهارس غير المستخدمة، نحتاج أيضا إلى معرفة ما إذا كانت هناك أية فهارس مكررة لأن الفهارس المكررة تستهلك أيضا مساحة غير ضرورية. في كثير من الأحيان، هناك حالات من الفهارس المعرفة على عمود جدول مع مفتاح فريد ويتم تعريف نفس العمود أيضا باسم المفتاح الأساسي. ومن شأن هذا الوضع أن يؤدي إلى وجود فهرس مكرر لأن المفتاح الأساسي هو نفسه "المفتاح الوحيد"، وفي هذه الحالة، ليست هناك حاجة إلى تحديد فهرس إضافي على نفس العمود كفهرس وحيد. يمكننا استخدام الاستعلام التالي لتحديد الفهارس المكررة في بوستجريسكل:

```
SELECT pg_size_pretty(sum(pg_relation_size(idx))::bigint) AS size,
(array_agg(idx))[1] AS idx1, (array_agg(idx))[2] AS idx2,(array_agg(idx))[3] AS
idx3, (array_agg(idx))[4] AS idx4
FROM (SELECT indexrelid::regclass AS idx, (indrelid::text ||E'\n'||
indclass::text ||E'\n'|| indkey::text ||E'\n'|| coalesce(indexprs::text,'')||
E'\n' || coalesce(indpred::text,'')) AS KEY
FROM pg_index) sub
GROUP BY KEY HAVING count(*)>1
ORDER BY sum(pg_relation_size(idx)) DESC;
```

وبمجرد تحديد الفهارس المكررة، يمكن بعد ذلك إسقاطها لاستعادة المساحة المفقودة.

يمكنك الرجوع إلى

https://gist.github.com/jberkus/6b1bcaf7724dfc2a54f3
http://www.databasesoup.com/2014/05/new-finding-unused-indexes-query.html

و التي تحتوي على مزيد من المعلومات المتعلقة بالفهارس غير المستخدمة.

فرض استخدام فهرس على استعلام

في هذه الوصفة، نعرض طرق مختلفة يمكن استخدامها لإجبار قاعدة البيانات على استخدام فهرس.

الاستعداد للعمل

عادة ما تكون وظيفة محسن بوستجريسكل هي تحديد من هو أكثر كفاءة المسح التسلسلي أو البحث الفهرسي عندما ينفذ إلى الجدول بواسطة استعلام لجلب النتائج. ومع ذلك، إذا قررنا أنه يستحق المراهنة على فهرس، فإنه يجب علينا تأكيد نتائجنا عن طريق اختبار تنفيذ الاستعلام في بيئة التطوير قبل نقل النتائج إلى بيئة الإنتاج.

كيف ينجز ذلك...

هناك طريقتان يمكننا من خلالها إجبار قاعدة البيانات على استخدام فهرس:

الأول هو عن طريق تعيين enable_seqscan إلى false.. ويمكن إثبات ذلك من خلال سيناريو معين على النحو التالي:

```
dvdrental=# create table test_no_index(id int);
dvdrental=# set enable_seqscan to false;
dvdrental=# explain select * from test_no_index where id > 12;
                          QUERY PLAN
---------------------------------------------------------------
```

```
Seq Scan on test_no_index  (cost=10000000000.00..10000000040.00 rows=800 width=4)
   Filter: (id > 12)
(2 rows)
```

- بعد ذلك، نقوم بإنشاء فهرس على الجدول المحدد لإعطاء محسن واحد أو أكثر من مسارات النفاذ:

```
dvdrental=# create index new_idx_test_no_index on test_no_
index(id);
CREATE INDEX
```

- إذا تحققنا الآن من خطة تنفيذ الاستعلام، سنرى أنه بدلا من المسح التسلسلي، تستخدم خطة الاستعلام البحث الفهرسي للوصول إلى الجدول لجلب نتيجة الاستعلام:

```
dvdrental=# explain select * from test_no_index where id > 12;
                         QUERY PLAN
------------------------------------------------------------
 Bitmap Heap Scan on test_no_index  (cost=10.35..30.35 rows=800 width=4)
   Recheck Cond: (id > 12)
   ->  Bitmap Index Scan on new_idx_test_no_index  (cost=0.00..10.15 rows=800 wi
dth=0)
         Index Cond: (id > 12)
(4 rows)
```

- طريقة أخرى هي تعيين قيمة معلمة التضبيط random_page_cost إلى قيمة أقل أو مكافئة ل seq_page_cost. من خلال القيام بذلك، سوف يفضل بوستجريسكل المسح الفهرسي لبعض استعلامات SQL. ويمكن القيام بذلك على النحو التالي:

```
dvdrental=# set random_page_cost = 2;
SET
```

كيف تعمل...

في القسم السابق، سيؤدي تعيين enable_seqscan إلى false إلى تعطيل عمليات المسح التسلسلي وإجبار المحسن على محاولة استخدام خطة مختلفة. في السيناريو الخاص بنا، قمنا بتعطيل المسح التفصيلي وإنشاء فهرس، new_idx_test_no_index، في الجدول test_no_index. من خلال القيام بذلك، قمنا بتقديم مسار وصول آخر للمحسن مع للجدول test_no_index.

وبالمثل، فإن خفض قيمة معلمة التضبيط random_page_cost سوف يدفع النظام إلى تفضيل المسح الفهرسي. افتراضيا، قيمة randon_page_cost هي ٤، وهي أعلى من القيمة الافتراضية لمعلمة التضبيط seq_page_cost، التي

هي ١، مما يسبب تفضيلا للمسح التسلسلي على المسح الفهرسي. قد يساعد خفض قيمة random_page_cost في بعض الاستعلامات حيث قد يفضل المحسن استخدام البحث الفهرسي.

تحديد استخدام القرص

في هذه الوصفة، سنقوم بعرض كمية استخدام القرص لقاعدة بيانات معينة والجداول المرتبطة بها والفهارس.

كيف ينجز ذلك...

يمكننا استخدام استعلام SQL التالي للعثور على الحجم الكلي لقاعدة بيانات في هذه الحالة موجودة قاعدة بيانات dvdrental:

```
dvdrental=# SELECT pg_size_pretty(pg_database_size('dvdrental')) As fulldbsize;

fulldbsize
------------
14 MB
(1 row)
```

في خرج الاستعلام هذا، يمكننا أن نرى أن الحجم الكلي لقاعدة البيانات dvdrental حوالي ١٤ ميغابايت.

وبالمثل، لعرض حجم الجداول الموجودة والفهارس المرتبطة بها في قاعدة بيانات dvdrental، يمكننا استخدام استعلام SQL التالي:

```
SELECT relname as "Table",
  pg_size_pretty(pg_relation_size(relid)) As " Table Size",
  pg_size_pretty(pg_total_relation_size(relid) -
  pg_relation_size(relid)) as "Index Size"
FROM pg_catalog.pg_statio_user_tables ORDER BY
pg_total_relation_size(relid) DESC;

        Table         |  Table Size | Index Size
---------------------+-------------+------------
 rental               | 1200 kB     | 1272 kB
 payment              | 864 kB      | 1368 kB
 film                 | 432 kB      | 256 kB
 film_actor           | 240 kB      | 296 kB
 inventory            | 200 kB      | 264 kB
```

```
 customer             | 72 kB       | 152 kB
 keytbl               | 40 kB       | 144 kB
 address              | 64 kB       | 88 kB
 city                 | 40 kB       | 88 kB
 film_category        | 48 kB       | 64 kB
 actor                | 16 kB       | 56 kB
 encdata              | 24 kB       | 32 kB
 store                | 8192 bytes  | 32 kB
 staff                | 8192 bytes  | 24 kB
 category             | 8192 bytes  | 16 kB
 country              | 8192 bytes  | 16 kB
 language             | 8192 bytes  | 16 kB
 test                 | 0 bytes     | 8192 bytes
 test_no_index        | 0 bytes     | 8192 bytes
 table_with_no_index  | 0 bytes     | 8192 bytes
(20 rows)
```

كيف تعمل...

إذا قمنا بفحص الناتج السابق، يمكننا أن نرى أسماء الجداول جنبا إلى جنب مع أحجام الجدول والفهرس. في الاستعلام السابق، استخدمنا وظيفتين pg_relation_size() و pg_total_relation_size ().

الوظيفة pg_relation_size() تبلغ عن حجم الجدول بالكيلوبايت والوظيفة pg_total_relation_size() تبلغ عن الحجم الكلي للجدول المستخدم على القرص بما في ذلك المساحة المستخدمة من قبل بيانات TOAST والفهارس. لذلك من أجل الحصول على أحجام الفهرس الصحيح لجميع الفهارس لجدول معين، طرحنا قيمة pg_relation_size () من (pg_total_relation_size باستخدام العمود relid كمعلمة في كل من الوظيفتين.

إذا كنت بحاجة إلى مزيد من المعلومات حول تحديد استخدام القرص، يمكنك الرجوع إلى http://wiki.postgresql.org/wiki/Disk_Usage و https://wiki.postgresql.org/wiki/Index_Maintenance.

هناك المزيد...

في هذا القسم، سوف نقدم بعض الروابط التي يمكن الرجوع إليها للحصول على المشورة بشأن التعامل مع قضايا الأداء المتعلقة ببوستجريسكل.

يمكنك الاطلاع على القائمة البريدية للأداء على

```
http://archives.postgresql.org/pgsql-performance/
```

يمكنك أيضا الرجوع إلى بعض روابط Wiki بوستجريسكل التي توضح ما يجب تضمينه في تقرير مشكلة الأداء وبعض معلومات تحري الخلل وإصلاحه المفيدة، على http://wiki.postgresql.org/wiki/Guide_to_reporting_problems و http://wiki.postgresql.org/wiki/Performance_Optimization

إذا كنت قد اشتريت دعم بريميوم من البائعين كـ 2ndQuadrant و EnterpriseDB ، يمكنك تسجيل تذاكر مع فريق الدعم فيما يتعلق بقضايا بوستجريسكل.

٧- التوفر العالي والنسخ

في هذا الفصل، سنغطي الوصفات التالية:

- إعداد تدفق النسخ الساخن
- النسخ باستخدام Slony-I
- النسخ باستخدام Londiste
- النسخ باستخدام Bucardo
- النسخ باستخدام DRBD
- إعداد عنقود Postgres-XC

مقدمة

إن أهم المميزات لأي قاعدة بيانات في بيئة الإنتاج هو التسامح مع الأخطاء، والتوفر مدى ٢٤/٧ ، والتكرار redundancy. ومن أجل هذا الغرض تملك بوستجريسكل عدة حلول للتوفر العالي وحلول النسخ والتكرار.

من منظور الأعمال، من المهم ضمان توافر البيانات على مدار الساعة طوال أيام الأسبوع في حالة حدوث كارثة أو تعطل قاعدة البيانات بسبب فشل القرص أو الجهاز. في مثل هذه الحالات، يصبح من الضروري التأكد من أن نسخة مكررة من البيانات متوفرة على خادم مختلف أو قاعدة بيانات مختلفة، بحيث يمكن تجاوز الفشل بسلاسة حتى عندما يكون الخادم الأساسي / قاعدة البيانات غير متوفرين.

في هذا الفصل، سوف نتحدث عن مختلف الحلول للتوفر العالي والنسخ، بما في ذلك بعض أدوات النسخ من خلال طرف ثالث مثل Slony-I ولونديستي Londiste وبوكاردو Bucardo. سوف نناقش أيضا النسخ على مستوى الكتلة باستخدام DRBD، وأخيرا، إنشاء عنقود بوستجريسكل موسع، وهذا هو، Postgres-XC.

إعداد تدفق النسخ الساخن

في هذه الوصفة، نحن في طريقنا لإقامة نسخ الفرعي / الرئيسي.

الاستعداد للعمل

لأجل هذا التمرين، سوف تحتاج إلى اثنين من أجهزة لينكس ، كل واحدة مع أحدث إصدار مثبت من بوستجريسكل.

سنستخدم عناوين IP التالية للخوادم الرئيسية والتابعة:

عنوان IP الرئيسي: ١٩٢.١٦٨.٠.٤

عنوان IP الفرعي: ١٩٢.١٦٨.٠.٥

قبل البدء في إعداد تدفق الفرعي-الرئيسي، فمن المهم أن يتم تضبيط الاتصال SSH بين الفرعي والرئيسي.

كيف ينجز ذلك...

قم بتنفيذ التسلسل التالي من الخطوات لإعداد تدفق النسخ الرئيسي والفرعي master-slave :

١. أولا، نذهب لإنشاء مستخدم على الرئيسي master، والذي سيستخدم من قبل الخادم الفرعي slave server للاتصال بقاعدة بيانات بوستجريسكل على الخادم الرئيسي:

```
psql -c "CREATE USER repuser REPLICATION LOGIN ENCRYPTED
PASSWORD 'charlie';"
```

٢. بعد ذلك، سنسمح للمستخدم المكرر replication الذي أنشئ في الخطوة السابقة للسماح بالنفاذ إلى الخادم الرئيسي بوستجريسكل.

عن طريق إجراء التغييرات اللازمة على النحو المذكور في ملف pg_hba.conf :

```
Vi  pg_hba.conf
Host    replication    repuser    192.168.0.5/32    md5
```

٣. في الخطوة التالية، سنقوم بتضبيط المعلمات في الملف postgresql.conf.

هذه المعلمات تحتاج إلى تعيين من أجل أن يعمل تدفق النسخ:

```
Vi /var/lib/pgsql/9.3/data/postgresql.conf

listen_addresses = '*'
wal_level = hot_standby
max_wal_senders = 3
wal_keep_segments = 8
archive_mode = on
archive_command = 'cp %p /var/lib/pgsql/archive/%f && scp %p
postgres@192.168.0.5:/var/lib/pgsql/archive/%f'

checkpoint_segments = 8
```

٤. بعد إجراء تغييرات المعلمة في ملف postgresql.conf في الخطوة السابقة، فإن الخطوة التالية هي إعادة تشغيل خادم بوستجريسكل في الخادم الرئيسي للسماح للتغييرات بأن تصبح نافذة المفعول:

```
pg_ctl -D /var/lib/pgsql/9.3/data restart
```

٥. قبل أن يستطيع الخادم الفرعي نسخ الرئيسي، سوف تحتاج إلى إعطائه قاعدة البيانات الأولية للبناء. ولهذه الغاية، سوف نقوم بإنشاء قاعدة بيانات النسخ الاحتياطي عن طريق نسخ دليل بيانات الخادم الأساسي إلى الوضع الاحتياطي. يجب تشغيل الأمر rsync كمستخدم جذر:

```
psql -U postgres -h 192.168.0.4 -c "SELECT pg_start_ backup('label', true)"
rsync -a /var/lib/pgsql/9.3/data/ 192.168.0.5:/var/lib/pgsql/9.3/ data/
--exclude postmaster.pid
psql -U postgres -h 192.168.0.4 -c "SELECT pg_stop_backup()"
```

٦. بمجرد ملء دليل البيانات المذكور في الخطوة السابقة، تكون الخطوة التالية هي تفعيل المعامل التالي في الملف postgresql.conf على الخادم الفرعي:

```
hot_standby = on
```

٧. الخطوة التالية ستكون نسخ الملف recovery.conf.sample في موقع $PGDATA على الخادم الفرعي ثم تضبيط المعلمات التالية:

```
cp /usr/pgsql-9.3/share/recovery.conf.sample
 /var/lib/pgsql/9.3/data/recovery.conf

standby_mode = on
primary_conninfo = 'host=192.168.0.4 port=5432 user=repuser
 password=charlie'
trigger_file = '/tmp/trigger.replication
restore_command = 'cp /var/lib/pgsql/archive/%f "%p"'
```

٧. الخطوة التالية ستكون لبدء الخادم الفرعي:

```
service postgresql-9.3 start
```

٨. الآن بعد تعيين خطوات النسخ المذكورة أعلاه، سوف نختبر النسخ. على الخادم الرئيسي، قم بتسجيل الدخول وإصدار أوامر SQL التالية:

```
psql -h 192.168.0.4  -d postgres -U postgres -W
postgres=# create database test;
postgres=# \c test;
test=# create table testtable ( testint int, testchar varchar(40) );
CREATE TABLE
test=# insert into testtable values ( 1, 'What A Sight.' );
INSERT 0 1
```

١٠. على الخادم الفرعي، سوف نقوم الآن بالتحقق مما إذا كانت قاعدة البيانات التي أنشئت حديثا والجدول الملائم، الذي أنشئ في الخطوة السابقة، تم نسخهما:

```
psql -h 192.168.0.5  -d test -U postgres -W
test=# select * from testtable;
testint | testchar
---------+--------------------------
1        | What  A Sight.
(1 row)
```

كيف تعمل...

وفيما يلي شرح للخطوات التي ذكرت في القسم السابق.

في الخطوة الأولى من القسم السابق، قمنا بإنشاء مستخدم يسمى repuser، والذي سيستخدم من قبل الخادم الفرعي لإجراء اتصال إلى الخادم الأساسي. في الخطوة الثانية من القسم السابق، قمنا بإجراء التغييرات الضرورية في ملف

pg_hba.conf للسماح بالنفاذ إلى الخادم الرئيسي من قبل الخادم الفرعي باستخدام معرف المستخدم الذي أنشئ في الخطوة ١. ثم قمنا بإجراء المعلمة اللازمة التغييرات على الرئيسي في الخطوة ٣ من القسم السابق لتضبيط تدفق النسخ. وفيما يلي وصف لهذه المعلمات:

- listen_addresses: تستخدم هذه المعلمة لتوفير عنوان IP المقترن بالواجهة التي تريد أن تستمع إليها بوستجريسكل. تشير قيمة * إلى جميع عناوين IP المتاحة.
- wal_level: تحدد هذه المعلمة مستوى التسجيل WAL المكتمل. حدد hot_standby لتدفق النسخ.
- wal_keep_segments: تحدد هذه المعلمة عدد ملفات التسجيل عبر الكتابة ١٦ ميغابايت التي سيحتفظ بها في دليل pg_xlog. القاعدة الأساسية هي أن المزيد من هذه الملفات قد تكون مطلوبة للتعامل مع نقطة تفتيش كبيرة.
- archive_mode: يؤدي تعيين هذه المعلمة إلى تفعيل إرسال شرائح التسجيل WAL المكتملة إلى وحدة التخزين في الأرشيف.
- archive_command: هذه المعلمة هي في الأساس أمر Shell ينفذ عند اكتمال قطعة التسجيل WAL . في حالتنا، نحن في الأساس ننسخ الملف إلى الجهاز المحلي ومن ثم نستخدم أمر النسخ الآمن لإرساله إلى الفرعي.
- max_wal_senders: تحدد هذه المعلمة العدد الإجمالي للاتصالات المتزامنة المسموح بها من خوادم الفرعي.
- checkpoint_segments: تحدد هذه المعلمة الحد الأقصى لعدد شرائح السجل بين نقاط التحقق التلقائية والتسجيل عبر الكتابة. وبمجرد إجراء تغييرات التضبيط الضرورية على الخادم الرئيسي، نقوم بإعادة تشغيل خادم بوستجريسكل على الرئيسي من أجل السماح لتغييرات التضبيط الجديدة أن تصبح نافذة المفعول. وهذا ما فعلناه في الخطوة ٤ من القسم السابق. في الخطوة ٥ من القسم السابق، نقوم أساسا ببناء الفرعي عن طريق نسخ دليل بيانات الخادم الأساسي إلى الفرعي.

الآن، وبعد توفير دليل البيانات المتاحة على الفرعي، فإن الخطوة التالية هي تضبيطه. وسنقوم بتغيير المعلمات الضرورية ذات الصلة بالنسخ على الخادم الفرعي في ملف postgresql.conf . نحدد الإعدادات التالية على الفرعي:

- hot_standby: تحدد هذه المعلمة ما إذا كان يمكنك الاتصال وتشغيل الاستعلامات عندما يكون الخادم في وضع استرداد الأرشيف أو وضع الاستعداد. في الخطوة التالية، نقوم بتضبيط الملف recovery.conf. هذا مطلوب أن يعد بحيث يمكن أن يبدأ الفرعي بتلقي سجلات من الرئيس. والمعلمات الموضحة التالية تضبط في ملف recovery.conf على الخادم الفرعي.

- standby_mode: هذه المعلمة، عند تفعيلها، تتسبب في عمل بوستجريسكل في وضع الاستعداد في تضبيط النسخ
- primary_conninfo: تحدد هذه المعلمة معلومات الاتصال التي يستخدمها الفرعي للاتصال بالرئيسي. بالنسبة للسيناريو لدينا، فقد عينا خادمنا الرئيسي كما ١٩٢.١٦٨.٠.٤ على المنفذ ٥٤٣٢ ونحن نستخدم المستخدم repuser مع كلمة المرور charlie لإجراء اتصال إلى الرئيس. تذكر أن repuser هو المستخدم الذي أنشئ في الخطوة الأولى من القسم السابق لهذا الغرض، وهذا هو، الاتصال إلى الرئيسي من الفرعي.
- trigger_file: عندما يضبط الخادم الفرعي في وضع الاستعداد، فإنه سيستمر في استعادة سجلات XLOG من الرئيسي. تحدد المعلمة trigger_file ما يستخدم لتشغيل الفرعي، لنقل وظائفه إلى وضع الاستعداد ودعمه كخادم أساسي أو رئيسي. عند هذه النقطة، أصبح الخادم الفرعي معدا تماما الآن ويمكننا أن نبدأ في تشغيله. ثم، تبدأ عملية النسخ. ويظهر ذلك في الخطوة ٨ من القسم السابق. في الخطوتين ٩ و ١٠ من القسم السابق، نحن ببساطة نختبر النسخ لدينا. نبدأ أولا بإنشاء قاعدة بيانات test ، ثم نقوم بالاتصال بقاعدة بيانات test وإنشاء جدول باسم testtable ، ثم نبدأ إدراج بعض السجلات في جدول testtable. الآن، هدفنا هو معرفة ما إذا كان قد نسخت هذه التغييرات على الفرعي. لاختبار هذا، نقوم بتسجيل الدخول إلى الفرعي على قاعدة بيانات test، ثم الاستعلام عن السجلات من جدول testtable كما هو موضح في الخطوة ١٠ من المقطع السابق. النتيجة النهائية التي نراها هي أن كافة السجلات التي عدلت/أدرجت على الخادم الأساسي مرئية على الفرعي، وبهذا يكون قد اكتمل تركيب وتضبيط تدفق النسخ.

يمكنك الرجوع إلى الروابط التالية للحصول على مزيد من المعلومات التفصيلية حول تدفق النسخ:

https://www.digitalocean.com/community/tutorials/how-to-set-upmaster-slave-replication-on-postgresql-on-an-ubuntu-12-04-vps
http://www.rassoc.com/gregr/weblog/2013/02/16/zero-topostgresql-streaming-replication-in-10-mins/

النسخ باستخدام Slony-I

هنا، نحن في طريقنا لإعداد النسخ باستخدام Slony-I . سنقوم بإعداد النسخ لبيانات الجدول بين قاعدتي بيانات اثنتين على نفس الخادم.

الاستعداد للعمل

نفذت الخطوات التي أجريت في هذه الوصفة على جهاز سينتوس الإصدار ٦. ومن المهم أيضا إزالة التعليمات المتعلقة بتدفق النسخ الساخن قبل إعداد النسخ باستخدام Slony-I.

سنحتاج أولا إلى تثبيت Slony-I. يجب تنفيذ الخطوات التالية من أجل تثبيت Slony-I:

١. أولا، اذهب إلى http://slony.info/downloads/2.2/source/ وحمل البرنامج.

٢. بمجرد الانتهاء من تحميل برنامج Slony-I، فإن الخطوة التالية هي فك ضغط الملف tar. ثم انتقل إلى الدليل الذي أنشئ حديثا. قبل القيام بذلك، يرجى التأكد من أن لديك حزمة postgresql-devel لنسخة بوستجريسكل الملائمة مثبة قبل تثبيت Slony-I:

```
tar xvfj slony1-2.2.3.tar.bz2

cd slony1-2.2.3
```

٣. في الخطوة التالية، نحن في طريقنا لتضبيط وتجميع وبناء البرنامج:

```
./configure --with-pgconfigdir=/usr/pgsql-9.3/bin/
make
make install
```

كيف ينجز ذلك...

تحتاج إلى تنفيذ تسلسل الخطوات التالية، من أجل نسخ البيانات بين جدولين باستخدام Slony-I النسخ:

١. أولا، ابدأ تشغيل خادم بوستجريسكل إذا لم تكن قد بدأت بالفعل:

```
pg_ctl -D $PGDATA start
```

٢. في الخطوة التالية، سنقوم بإنشاء قاعدتي بيانات، test1 و test2، واللتان ستستخدمان كقاعدة بيانات المصدر والهدف على التوالي:

```
createdb test1
createdb test2
```

٣. في الخطوة التالية، سوف نقوم بإنشاء الجدول t_test على قاعدة بيانات المصدر، test1، وإدراج بعض السجلات في ذلك:

```
psql -d test1
```

```
test1=# create table t_test (id numeric primary key, name varchar);
test1=# insert into t_test values(1,'A'),(2,'B'), (3,'C');
```

٤. سنقوم الآن بإعداد قاعدة البيانات المستهدفة عن طريق نسخ تعريفات الجدول من قاعدة بيانات المصدر test1:

```
pg_dump -s -p 5432 -h localhost test1 | psql -h localhost -p 5432 test2
```

٥. سوف نقوم الآن بالاتصال بقاعدة البيانات المستهدفة، test2، والتحقق من عدم وجود بيانات في جداول قاعدة البيانات test2:

```
psql -d  test2
test2=# select * from t_test;
```

٦. سنقوم الآن بإعداد سكريبت slonik للتابع والرئيسي، وهذا هو مصدر / هدف، الإعداد في هذا السيناريو، حيث أننا ننسخ بين قاعدتي بيانات مختلفتين على نفس الخادم، الخيار الوحيد لسلسة الاتصال المختلفة سيكون اسم قاعدة البيانات:

```
cd /usr/pgsql-9.3/bin
vi init_master.slonik
```

```
#!/bin/sh
cluster name = mycluster;
node 1 admin conninfo = 'dbname=test1 host=localhost
port=5432 user=postgres password=postgres';
node 2 admin conninfo = 'dbname=test2 host=localhost
port=5432 user=postgres password=postgres';
init cluster ( id=1);
create set (id=1, origin=1);
set add table(set id=1, origin=1, id=1, fully qualified
name = 'public.t_test');
store node (id=2, event node = 1);
store path (server=1, client=2, conninfo='dbname=test1
host=localhost port=5432 user=postgres password=postgres');
store path (server=2, client=1, conninfo='dbname=test2
host=localhost port=5432 user=postgres password=postgres');
store listen (origin=1, provider = 1, receiver = 2);
store listen (origin=2, provider = 2, receiver = 1);
```

٧. سنقوم الآن بإنشاء سيناريو slonik للاشتراك في الفرعي، وهذا هو الهدف:

```
cd /usr/pgsql-9.3/bin
vi init_slave.slonik
```

```
#!/bin/sh
cluster name = mycluster;
node 1 admin conninfo = 'dbname=test1 host=localhost
port=5432 user=postgres password=postgres';
node 2 admin conninfo = 'dbname=test2 host=localhost
port=5432 user=postgres password=postgres';
subscribe set ( id = 1, provider = 1, receiver = 2, forward
= no);
```

٨. سنقوم الآن بتشغيل السكريبت init_master.slonik الذي أنشئ في الخطوة ٦ تشغيل هذا على الرئيسي، كما يلي:

```
cd /usr/pgsql-9.3/bin
slonik init_master.slonik
```

٩. الآن سنشغل السكريبت init_slave.slonik : الذي أنشئ في الخطوة ٧ وتشغيل هذا على الفرعي، على النحو التالي:

```
cd /usr/pgsql-9.3/bin
slonik init_slave.slonik
```

١٠. في الخطوة التالية, سنبدأ البرنامج الخفي SLON الرئيسي:

```
nohup slon mycluster "dbname=test1 host=localhost port=5432
user=postgres password=postgres" &
```

١١. في الخطوة التالية, سنبدأ البرنامج الخفي SLON الفرعي:

```
nohup slon mycluster "dbname=test2 host=localhost port=5432
user=postgres password=postgres" &
```

١٢. بعد ذلك، سوف نقوم بالاتصال بالرئيس، وهذا هو، قاعدة بيانات مصدر test1, وإدراج بعض السجلات في الجدول t_test:

```
psql -d test1
test1=# insert into t_test values (5,'E');
```

١٣. سنقوم الآن باختبار النسخ عن طريق تسجيل الدخول إلى الفرعي، أي قاعدة بيانات الهدف test2، ونرى ما إذا كانت السجلات المدرجة في جدول t_test مرئية:

```
psql -d test2
test2=# select * from t_test;
 id | name
----+------
  1 | A
  2 | B
  3 | C
  5 | E
(4 rows)
```

كيف تعمل...

سنناقش الآن الخطوات التي أجريت في القسم السابق:

- في الخطوة ١، نبدأ بتشغيل أولا خادم بوستجريسكل إذا لم يكن قد بدأ بالفعل. في الخطوة ٢، نقوم بإنشاء قاعدتي بيانات، هما test1 و test2، والتي ستكون بمثابة قواعد بيانات المصدر (الرئيسي) والهدف (الفرعي).
- في الخطوة ٣، نقوم بتسجيل الدخول إلى قاعدة بيانات المصدر test1، وإنشاء جدول اختبار t_test ، وإدراج بعض السجلات في الجدول.
- في الخطوة ٤، أنشأنا قاعدة البيانات الهدف، test2، عن طريق نسخ تعريفات الجدول الموجودة في قاعدة البيانات المصدر وتحميلها في test2 باستخدام الأداة المساعدة pg_dump
- في الخطوة ٥، نقوم بتسجيل الدخول إلى قاعدة البيانات المستهدفة،test2 والتحقق من عدم وجود سجلات موجودة في الجدول t_test لأنه في الخطوة ٤ استخرجنا تعريفات الجدول فقط في قاعدة البيانات test2 من قاعدة البيانات test1.
- في الخطوة ٦، أنشأنا سكربت slonik لإعداد النسخ بين الرئيس والفرعي. في الملف init_master.slonik، أولا عرفنا اسم العنقود باسم mycluster. ثم نقوم بتعريف العقد في العنقود. سيكون لكل عقدة رقم مرتبط بسلسلة اتصال تحتوي على معلومات الاتصال. وتعرف مدخلة العقدة لكل من قواعد البيانات المصدر والهدف. أوامر store_path ضرورية، بحيث تعرف كل عقدة كيفية التواصل مع الأخرى.

- في الخطوة ٧، أنشأنا سكربت slonik للتسجيل في الفرعي، وهذه هي، قاعدة البيانات الهدف test2. مرة أخرى، يحتوي السكربت على معلومات مثل اسم العنقود وإدخالات العقدة التي عين لها رقم خاص متصل بمعلومات سلسلة الاتصال. كما أنه يحتوي على مجموعة المشتركين.
- في الخطوة ٨، نشغل الملف init_master.slonik على الرئيسي. وبالمثل، في الخطوة ٩، نشغل الملف init_slave.slonik على الفرعي.
- في الخطوة ١٠، نبدأ تشغيل البرنامج الخفي SLON الرئيسي. في الخطوة ١١، نبدأ البرنامج الخفي SLON الفرعي.
- وتستخدم الخطوات اللاحقة، ١٢ و ١٣، لاختبار النسخ. لهذا الغرض، في الخطوة ١٢ من القسم السابق، نقوم أولا بتسجيل الدخول إلى قاعدة بيانات المصدر test1 وإدراج بعض السجلات في الجدول t_test. للتحقق مما إذا ما قد نسخت السجلات التي أدرجت حديثا في قاعدة البيانات المستهدفة، test2، وقد قمنا بتسجيل الدخول إلى قاعدة البيانات test2 في الخطوة ١٣. مجموعة النتائج التي حصل عليها من إخراج الاستعلام يؤكد أن السجلات التي جرى تغييرها / إدراجها في الجدول t_test نسخت في قاعدة البيانات test1 بنجاح عبر قاعدة البيانات المستهدفة، test2.

لمزيد من المعلومات حول نسخ الـ Slony-I ، قم بزيارة:

http://slony.info/documentation/tutorial.html

هناك المزيد...

إذا كنت تستخدم Slony-I للنسخ بين اثنين من خوادم مختلفة، بالإضافة إلى الخطوات المذكورة في القسم كيف ينجز ذلك...، سيكون عليك أيضا تفعيل معلومات المصادقة في ملف pg_hba.conf الموجودة على كل من خوادم المصدر والهدف على سبيل المثال، لنفترض أن IP خادم المصدر هو ١٩٢.١٦٨.١٦.٤٤ و IP خادم الهدف هو ١٩٢.١٦٨.١٦.٥٦ ونحن نستخدم مستخدم super لنسخ البيانات.

إذا كان هذا هو الوضع، سيكون علينا إدخال المعلومات في ملف المصدر pg_hba.conf ، على النحو التالي:

```
host         postgres              super          192.168.16.44/32            md5
```

وبالمثل، في ملف خادم الهدف pg_hba.conf، سيكون علينا إدخال معلومات التوثيق، على النحو التالي:

```
host          postgres              super       192.168.16.56/32              md5
```

كذلك في سكريبت shell الذي استخدم في Slony-I، فمهما كانت معلومات الاتصال للمضيف هو مضيف محلي localhost، فإن المدخلة بحاجة أن تستبدل بعناوين IP خوادم المصدر والهدف.

النسخ باستخدام Londiste

في هذه الوصفة، سنعرض لك كيفية نسخ البيانات باستخدام Londiste.

الاستعداد للعمل

لهذا الإعداد، نستخدم نفس الجهاز المضيف سينتوس لينكس لنسخ البيانات بين قواعد بيانات اثنين. ويمكن أيضا أن يعد هذا باستخدام اثنين من أجهزة لينكس المنفصلة تعمل على Vmware أو VirtualBox أو أي برنامج وهمي آخر. ومن المفترض أن تثبت أحدث إصدار من بوستجريسكل، الإصدار ٩.٣. استخدمنا سينتوس الإصدار ٦ كنظام التشغيل لينكس لهذا التمرين.

لإعداد النسخ Londiste على جهاز لينكس، قم بالخطوات التالية:

١. اذهب إلى

```
http://pgfoundry.org/projects/skytools/
```

وحمل أحدث إصدار من Skytools 3.2، وهذا هو، tarball skytools-3.2.tar.gz.

٢. استخرج ملف tarball، على النحو التالي:

```
tar -xvzf skytools-3.2.tar.gz
```

٣. انتقل إلى الموقع الجديد وقم ببناء وتجميع البرنامج:

```
cd skytools-3.2
./configure  --prefix=/var/lib/pgsql/9.3/Sky –with-pgconfig=/usr/ pgsql-9.3/bin/pg_config

make

make instal
```

٤. أيضا، قم بتعيين متغير البيئة PYTHONPATH، كما هو موضح هنا. وبدلا من ذلك، يمكنك أيضا تعيينه في السكريبت .bash_profile :

```
export PYTHONPATH=/opt/PostgreSQL/9.2/Sky/lib64/python2.6/sitepackages/
```

كيف ينجز ذلك...

١. سنقوم بتنفيذ التسلسل التالي من الخطوات لإعداد النسخ بين قواعد البيانات المختلفة باستخدام Londiste.

أولا، يجب إنشاء قاعدتي البيانات التي سيحدث بينها النسخ:

```
createdb node1
createdb node2
```

٢. تعبئة قاعدة البيانات node1 بالبيانات باستخدام الأداة المساعدة pgbench :

```
pgbench -i -s 2 -F 80 node1
```

٣. قم بإضافة أي مفتاح أساسي ومفاتيح خارجية للجداول في قاعدة البيانات node1 المطلوبة للنسخ المتماثل.

قم بإنشاء ملف sql. التالي وإضافة السطور التالية إليه:

```
vi /tmp/prepare_pgbenchdb_for_londiste.sql -- add primary key to history table
```

```
ALTER TABLE pgbench_history ADD COLUMN hid SERIAL PRIMARY KEY;
-- add foreign keys
ALTER TABLE pgbench_tellers ADD CONSTRAINT pgbench_tellers_ branches_fk FOREIGN
KEY(bid) REFERENCES pgbench_branches;
ALTER TABLE pgbench_accounts ADD CONSTRAINT pgbench_accounts_ branches_fk
FOREIGN KEY(bid) REFERENCES pgbench_branches;
ALTER TABLE pgbench_history ADD CONSTRAINT pgbench_history_ branches_fk FOREIGN
KEY(bid) REFERENCES pgbench_branches;
ALTER TABLE pgbench_history ADD CONSTRAINT pgbench_history_ tellers_fk FOREIGN
KEY(tid) REFERENCES pgbench_tellers;
ALTER TABLE pgbench_history ADD CONSTRAINT pgbench_history_ accounts_fk FOREIGN
KEY(aid) REFERENCES pgbench_accounts;
```

٤. سنقوم الآن بتحميل ملف sql. الذي أنشئ في الخطوة السابقة وتحميله في قاعدة البيانات:

```
psql node1 -f /tmp/prepare_pgbenchdb_for_londiste.sql
```

٥. سنقوم الآن بتعبئة قاعدة بيانات node2 بتعريفات الجدول من جداول قاعدة البيانات node1:

```
pg_dump -s -t 'pgbench*' node1 > /tmp/tables.sql
psql  -f /tmp/tables.sql node2
```

٦. الآن تبدأ عملية النسخ. سنقوم أولا بإنشاء ملف التضبيط londiste.ini مع المعلمات التالية من أجل إعداد عقدة الجذر لقاعدة بيانات المصدر، node1:

```
vi londiste.ini
```

```
[londiste3]
job_name = first_table
db = dbname=node1
queue_name = replication_queue
logfile = /home/postgres/log/londiste.log
pidfile = /home/postgres/pid/londiste.pid
```

٧. في الخطوة التالية، سنقوم باستخدام ملف التضبيط londiste.ini الذي أنشئ في الخطوة السابقة لإعداد عقدة الجذر لقاعدة بيانات node1 كما هو موضح هنا:

```
[postgres@localhost bin]$ ./londiste3 londiste3.ini create-root node1
dbname=node1
2014-12-09 18:54:34,723 2335 WARNING No host= in public connect
string, bad idea
2014-12-09 18:54:35,210 2335 INFO plpgsql is installed
2014-12-09 18:54:35,217 2335 INFO pgq is installed
2014-12-09 18:54:35,225 2335 INFO pgq.get_batch_cursor is
installed
2014-12-09 18:54:35,227 2335 INFO pgq_ext is installed
2014-12-09 18:54:35,228 2335 INFO pgq_node is installed
2014-12-09 18:54:35,230 2335 INFO londiste is installed
2014-12-09 18:54:35,232 2335 INFO londiste.global_add_table is
installed
2014-12-09 18:54:35,281 2335 INFO Initializing node
2014-12-09 18:54:35,285 2335 INFO Location registered
2014-12-09 18:54:35,447 2335 INFO Node "node1" initialized for
queue "replication_queue" with type "root"
2014-12-09 18:54:35,465 2335 INFO Don
```

٨. سنقوم الآن بتشغيل worker الخفي لعقدة الجذر:

```
[postgres@localhost bin]$ ./londiste3 londiste3.ini worker
2014-12-09 18:55:17,008 2342 INFO Consumer uptodate = 1
```

٩. في الخطوة التالية، سوف نقوم بإنشاء ملف التضبيط slave.ini من أجل إنشاء عقدة ورقة لقاعدة بيانات الهدف :node2

```
vi slave.ini
[londiste3]
job_name = first_table_slave
db = dbname=node2
queue_name = replication_queue
logfile = /home/postgres/log/londiste_slave.log
pidfile = /home/postgres/pid/londiste_slave.pid
```

١٠. سنقوم الآن بتهيئة العقدة في قاعدة البيانات المستهدفة:

```
./londiste3 slave.ini create-leaf node2 dbname=node2 –
provider=dbname=node1
2014-12-09 18:57:22,769 2408 WARNING No host= in public connect
string, bad idea
2014-12-09 18:57:22,778 2408 INFO plpgsql is installed
2014-12-09 18:57:22,778 2408 INFO Installing pgq
2014-12-09 18:57:22,778 2408 INFO Reading from /var/lib/
pgsql/9.3/Sky/share/skytools3/pgq.sql
2014-12-09 18:57:23,211 2408 INFO pgq.get_batch_cursor is
installed
2014-12-09 18:57:23,212 2408 INFO Installing pgq_ext
2014-12-09 18:57:23,213 2408 INFO Reading from /var/lib/
pgsql/9.3/Sky/share/skytools3/pgq_ext.sql
2014-12-09 18:57:23,454 2408 INFO Installing pgq_node
2014-12-09 18:57:23,455 2408 INFO Reading from /var/lib/
pgsql/9.3/Sky/share/skytools3/pgq_node.sql
2014-12-09 18:57:23,729 2408 INFO Installing londiste
2014-12-09 18:57:23,730 2408 INFO Reading from /var/lib/
pgsql/9.3/Sky/share/skytools3/londiste.sql
2014-12-09 18:57:24,391 2408 INFO londiste.global_add_table is
installed
2014-12-09 18:57:24,575 2408 INFO Initializing node
2014-12-09 18:57:24,705 2408 INFO Location registered
2014-12-09 18:57:24,715 2408 INFO Location registered
2014-12-09 18:57:24,744 2408 INFO Subscriber registered: node2
2014-12-09 18:57:24,748 2408 INFO Location registered
2014-12-09 18:57:24,750 2408 INFO Location registered
2014-12-09 18:57:24,757 2408 INFO Node "node2" initialized for
queue "replication_queue" with type "leaf"
2014-12-09 18:57:24,761 2408 INFO Done
```

١١. سنقوم الآن بإطلاق worker الخفي لقاعدة البيانات المستهدفة، وهذا هو،node 2:

```
[postgres@localhost bin]$ ./londiste3 slave.ini worker
2014-12-09 18:58:53,411 2423 INFO Consumer uptodate = 1
```

١٢. سنقوم الآن بإنشاء ملف التضبيط، وهو pgqd.ini للبرنامج ticker الخفي:

```
vi pgqd.ini
[pgqd]
logfile = /home/postgres/log/pgqd.log
pidfile = /home/postgres/pid/pgqd.pid
```

١٣. باستخدام ملف التضبيط الذي أنشئ في الخطوة السابقة، سوف نطلق البرنامج ticker الخفي :

```
[postgres@localhost bin]$ ./pgqd pgqd.ini
2014-12-09 19:05:56.843 2542 LOG Starting pgqd 3.2
2014-12-09 19:05:56.844 2542 LOG auto-detecting dbs ...
2014-12-09 19:05:57.257 2542 LOG node1: pgq version ok: 3.2
2014-12-09 19:05:58.130 2542 LOG node2: pgq version ok: 3.2
```

١٤.سنقوم الآن بإضافة كافة الجداول إلى النسخ على العقدة الجذر:

```
[postgres@localhost bin]$ ./londiste3 londiste3.ini add-table --all
2014-12-09 19:07:26,064 2614 INFO Table added: public.pgbench_accounts
2014-12-09 19:07:26,161 2614 INFO Table added: public.pgbench_branches
2014-12-09 19:07:26,238 2614 INFO Table added: public.pgbench_history
2014-12-09 19:07:26,287 2614 INFO Table added: public.pgbench_tellers
```

١٥ - وبالمثل، تضاف جميع الجداول إلى النسخ على عقدة الورقة:

```
[postgres@localhost bin]$ ./londiste3 slave.ini add-table —all
```

١٦. سنقوم الآن بتوليد بعض الزيارات على قاعدة بيانات مصدر node1:

```
pgbench -T 10 -c 5 node1
```

١٧. سنستخدم الآن أداة المقارنة compare المتوفرة مع الأمر londiste3 للتحقق من الجداول في كل العقد. أي قاعدة بيانات المصدر (node1) وقاعدة بيانات الوجهة (node2) لها نفس كمية البيانات:

```
[postgres@localhost bin]$ ./londiste3 slave.ini compare

2014-12-09 19:26:16,421 2982 INFO Checking if node1 can be used for copy
2014-12-09 19:26:16,424 2982 INFO Node node1 seems good source, using it
```

```
2014-12-09 19:26:16,425 2982 INFO public.pgbench_accounts: Using node node1
as provider
2014-12-09 19:26:16,441 2982 INFO Provider: node1 (root)
2014-12-09 19:26:16,446 2982 INFO Locking public.pgbench_accounts
2014-12-09 19:26:16,447 2982 INFO Syncing public.pgbench_accounts
2014-12-09 19:26:18,975 2982 INFO Counting public.pgbench_accounts
2014-12-09 19:26:19,401 2982 INFO srcdb: 200000 rows, checksum=167607238449

2014-12-09 19:26:19,706 2982 INFO dstdb: 200000 rows, checksum=167607238449

2014-12-09 19:26:19,715 2982 INFO Checking if node1 can be used for copy
2014-12-09 19:26:19,716 2982 INFO Node node1 seems good source, using it
2014-12-09 19:26:19,716 2982 INFO public.pgbench_branches: Using node node1 as
provider
2014-12-09 19:26:19,730 2982 INFO Provider: node1 (root)
2014-12-09 19:26:19,734 2982 INFO Locking public.pgbench_branches
2014-12-09 19:26:19,734 2982 INFO Syncing public.pgbench_branches
2014-12-09 19:26:22,772 2982 INFO Counting public.pgbench_branches
2014-12-09 19:26:22,804 2982 INFO srcdb: 2 rows, checksum=-3078609798
2014-12-09 19:26:22,812 2982 INFO dstdb: 2 rows, checksum=-3078609798
2014-12-09 19:26:22,866 2982 INFO Checking if node1 can be used for copy
2014-12-09 19:26:22,877 2982 INFO Node node1 seems good source, using it
2014-12-09 19:26:22,878 2982 INFO public.pgbench_history: Using node node1 as
provider
2014-12-09 19:26:22,919 2982 INFO Provider: node1 (root)
2014-12-09 19:26:22,931 2982 INFO Locking public.pgbench_history
2014-12-09 19:26:22,932 2982 INFO Syncing public.pgbench_history
2014-12-09 19:26:25,963 2982 INFO Counting public.pgbench_history
2014-12-09 19:26:26,008 2982 INFO srcdb: 715 rows, checksum=9467587272
2014-12-09 19:26:26,020 2982 INFO dstdb: 715 rows, checksum=9467587272
2014-12-09 19:26:26,056 2982 INFO Checking if node1 can be used for copy
2014-12-09 19:26:26,063 2982 INFO Node node1 seems good source, using it
2014-12-09 19:26:26,064 2982 INFO public.pgbench_tellers: Using node node1 as
provider
2014-12-09 19:26:26,100 2982 INFO Provider: node1 (root)
2014-12-09 19:26:26,108 2982 INFO Locking public.pgbench_tellers
2014-12-09 19:26:26,109 2982 INFO Syncing public.pgbench_tellers
2014-12-09 19:26:29,144 2982 INFO Counting public.pgbench_tellers
2014-12-09 19:26:29,176 2982 INFO srcdb: 20 rows, checksum=4814381032
2014-12-09 19:26:29,182 2982 INFO dstdb: 20 rows, checksum=4814381032
```

كيف تعمل...

فيما يلي شرح للخطوات المتخذة في القسم السابق:

- في البداية، في الخطوة ١، نقوم بإنشاء قاعدتي بيانات، هي node1 و node2، التي تستخدم كقواعد البيانات المصدر والهدف، على التوالي، من منظور النسخ. في الخطوة ٢، نقوم بملء قاعدة البيانات node1 باستخدام الأداة المساعدة pgbench .
- في الخطوة ٣ من القسم السابق، نضيف ونحدد المفتاح الأساسي منها، وعلاقات المفاتيح الخارجية على الجداول المختلفة ووضع هذه الأوامر DDL في ملف .SQL.
- في الخطوة ٤، نقوم بتنفيذ الأوامر DDL المذكورة في الخطوة ٣ على قاعدة البيانات node1؛ وبالتالي، وبهذه الطريقة، نحن نفرض المفتاح الأساسي وتعريفات المفتاح الخارجي على الجداول في مخطط pgbench في قاعدة البيانات node1.
- في الخطوة ٥، نستخرج تعريفات الجدول من الجداول في مخطط pgbench في قاعدة البيانات NODE1 وتحميل هذه التعريفات في قاعدة البيانات NODE2. وسوف نناقش الآن الخطوات من ٦ إلى ٨ من الفرع السابق.
- في الخطوة ٦، نقوم بإنشاء ملف التضبيط الذي سيستخدم بعد ذلك في الخطوة ٧ لإنشاء عقدة الجذر لقاعدة بيانات المصدر node1.
- في الخطوة ٨، سنطلق worker الخفي للعقدة الجذر. فيما يتعلق بالإدخالات المذكورة في ملف التضبيط في الخطوة ٦، نحدد أولا عملا يجب أن يكون له اسم، بحيث يمكن التعرف على العمليات المميزة بسهولة. ثم نقوم بتعريف سلسلة اتصال بمعلومات للاتصال بقاعدة بيانات المصدر، وهو node1 ثم نقوم بتعريف اسم قائمة انتظار النسخ المعنية. وأخيرا، فإننا نحدد موقع log وملفات pid.
- سوف نناقش الآن الخطوات من ٩ إلى ١١ من الفرع السابق. في الخطوة ٩، نحدد ملف التضبيط، الذي يستخدم بعد ذلك في الخطوة ١٠ لإنشاء عقدة ورقة لقاعدة البيانات الهدف، وهو node2.
- في الخطوة ١١، نطلق البرنامج worker الخفي لعقد ورقة. تحتوي الإدخالات في ملف التضبيط في الخطوة ٩ على سلسلة الاتصال job_name من أجل الاتصال بقاعدة البيانات المستهدفة، التي هي node2، واسم قائمة انتظار النسخ المتماثل المعنية، وموقع log و pid المعنية. إن الدور الرئيسي في الخطوة ١١ يقوم به الفرعي، وهي قاعدة البيانات الهدف، حيث تجد الرئيسي أو المزود، وهي هنا قاعدة بيانات المصدر node1.

- سنتحدث الآن عن الخطوتين ١٢ و ١٣ من القسم السابق. في الخطوة ١٢، نحدد تضبيط ticker ، وبمساعدتها نطلق عملية ticker المذكورة في الخطوة ١٣. عندما يبدأ ticker الخفي بنجاح، يصبح لدينا جميع المكونات وعمليات الإعداد واللازمة للنسخ. ومع ذلك، لم نحدد بعد ما الأشياء التي بحاجة للنسخ.
- في الخطوة ١٤ و ١٥، نقوم بتعريف الجداول لتنسخ على كل من قواعد البيانات المصدر والهدف، والتي هي node1 و node2 ، على التوالي.
- وأخيرا، سنتحدث عن الخطوتين ١٦ و ١٧ من القسم السابق. هنا، في هذه المرحلة، نقوم باختبار النسخ الذي أعددناه بين قاعدة بيانات المصدر node1 وقاعدة بيانات الهدف node2.
- في الخطوة ١٦، نولد بعض الزيارات على قاعدة البيانات المصدر node1 عن طريق تشغيل pgbench مع خمسة اتصالات متوازية بقاعدة البيانات وتوليد حركة المرور لمدة ١٠ ثانية.
- في الخطوة ١٧، نتحقق مما إذا كانت الجداول الموجودة على كل من قواعد بيانات المصدر والهدف لها نفس البيانات. لهذا الغرض، نستخدم الأمر compare على مزود ومسجل العقد المشترك ثم العد والمجموع الاختباري للصفوف على كلا الجانبين. الإخراج الجزئي من القسم السابق يخبرك أن البيانات نسخت بنجاح بين كافة الجداول التي هي جزء من النسخ إعداد بين قاعدة بيانات المصدر node1 وقاعدة البيانات الهدف node2، أيضا العد والمجموع الاختباري من الصفوف لجميع الجداول على قواعد البيانات المصدر والهدف هي متطابقة:

```
2014-12-09 19:26:18,975 2982 INFO Counting public.pgbench_accounts
2014-12-09 19:26:19,401 2982 INFO srcdb: 200000 rows,
checksum=167607238449
2014-12-09 19:26:19,706 2982 INFO dstdb: 200000 rows,
checksum=167607238449

2014-12-09 19:26:22,772 2982 INFO Counting public.pgbench_branches
2014-12-09 19:26:22,804 2982 INFO srcdb: 2 rows,
checksum=-3078609798

2014-12-09 19:26:22,812 2982 INFO dstdb: 2 rows,
checksum=-3078609798
2014-12-09 19:26:25,963 2982 INFO Counting public.pgbench_history
2014-12-09 19:26:26,008 2982 INFO srcdb: 715 rows,
checksum=9467587272
2014-12-09 19:26:26,020 2982 INFO dstdb: 715 rows,
checksum=9467587272

2014-12-09 19:26:29,144 2982 INFO Counting public.pgbench_tellers
2014-12-09 19:26:29,176 2982 INFO srcdb: 20 rows,
```

```
checksum=4814381032
2014-12-09 19:26:29,182 2982 INFO dstdb: 20 rows,
checksum=4814381032
```

تحقق من الروابط التالية لمزيد من المعلومات حول النسخ Londiste

https://wiki.postgresql.org/wiki/Londiste_Tutorial_(Skytools_2)
http://manojadinesh.blogspot.in/2012/11/skytools-londistereplication.html

النسخ باستخدام Bucardo

في هذه الوصفة، سنعرض لك النسخ بين اثنين من قواعد البيانات باستخدام Bucardo.

الاستعداد للعمل

نفذ هذا التمرين على حاسوب Red Hat لينكس.

قم بتثبيت حزمة EPEL لمنصة Red Hat من :

https://fedoraproject.org/ wiki/EPEL

ثم قم بتثبيت هذه RPMs مع الأمر yum التالي:

```
yum install perl-DBI perl-DBD-Pg perl-DBIx-Safe
```

إذا لم يكن مثبتا مسبقا، قم بتنزيل مستودع بوستجريسكل من:

http://yum.pgrpms.org/repopackages.php.

بعد ذلك، قم بتثبيت الحزمة التالية. وهذا مطلوب لأن Bucardo مكتوب في بيرل Perl:

```
yum install postgresql93-plperl
```

- لتثبيت Bucardo حمل أحدث نسخة من برنامج Bucardo وهو Bucardo نسخة ٥.٢.٠، من:

http://bucardo.org/wiki/Bucardo.

- قم بفك ملف tarball ، ثم انتقل إلى المجلد الذي أنشئ حديثا، واجمع وابني البرنامج:

```
tar xvfz Bucardo-5.2.0.tar.gz
cd Bucardo-5.2.0
perl Makefile.PL
make
```

```
make install
```

كيف ينجز ذلك...

وفيما يلي تسلسل كامل من الخطوات التي استخدمت لضبط النسخ بين اثنين من قواعد البيانات باستخدام Bucardo:

١. الخطوة الأولى هي تثبيت Bucardo. وهذا هو، إنشاء قاعدة بيانات Bucardo الرئيسية التي تحتوي على المعلومات التي سوف يحتاجها Bucardo الخفي :

```
[postgres@localhost ~]$ bucardo install --batch —quiet
```

٢. إنشاء مستخدم bucardo ذي الصلاحيات الموسعة. في الخطوة التالية، نقوم بإنشاء قواعد بيانات المصدر والهدف، أي gamma1 و gamma2 على التوالي، والتي تحتاج إلى النسخ:

```
[postgres@localhost ~]$ psql -qc 'create database gamma1'
psql -d gamma1-qc 'create table t1 (id serial primary key, email text)'
[postgres@localhost ~]$ psql -qc 'create database gamma2 template gamma1'
```

٣. في الخطوة التالية، نحن نبلغ Bucardo حول قواعد البيانات التي سوف تشارك في النسخ:

```
postgres@localhost ~]$ bucardo add db db1 dbname=gamma1
Added database "db1
[postgres@localhost ~]$ bucardo add db db2 dbname=gamma2
Added database "db2"
```

٤. بعدها، ننشئ قطيع myherd وتشمل تلك الجداول من قواعد البيانات المصدر التي ستكون جزءا من إعداد النسخ:

```
[postgres@localhost ~]$ bucardo add herd myherd t1
Created relgroup "myherd"
Added the following tables or sequences:
public.t1 (DB: db1)
The following tables or sequences are now part of the relgroup
"myherd":
public.t1
```

٥. في الخطوة التالية، ننشئ مزامنة المصدر :

```
[postgres@localhost ~]$ bucardo add sync beta herd=myherd
dbs=db1:source
```

```
Added sync "beta"
Created a new dbgroup named "beta"
```

٦. بعد ذلك، نقوم بإنشاء مزامنة الهدف:

```
[postgres@localhost ~]$ bucardo add sync charlie herd=myherd
dbs=db1:source,db2:target
Added sync "charlie"
Created a new dbgroup named "charlie"
```

٧. في هذه المرحلة، نكون قد أعددنا إجراء النسخ ، وبالتالي فإن الخطوة التالية هي بدء خدمة Bucardo :

```
[postgres@localhost ~]$ bucardo start
Checking for existing processes
Removing file "pid/fullstopbucardo"
Starting Bucardo
```

٨. الخطوة التالية هي اختبار إعداد النسخ. لهذا الغرض، سنقوم بإدراج بعض السجلات في الجدول t1 على قاعدة بيانات مصدر gamma1:

```
psql  -d  gamma1
gamma1=# insert into t1 values (1,'wallsingh@gmail.com');
INSERT 0 1
gamma1=# insert into t1 values (2,'neha.verma@gmail.com');
INSERT 0 1
```

٩. الآن بعد أن قمنا بإدراج بعض السجلات في قاعدة بيانات المصدر في الخطوة السابقة، نحن بحاجة إلى التحقق من نسخ هذه التغييرات في قاعدة بيانات الهدف gamma2:

```
psql -d gamma2
gamma2=# select * from t1;
 id |        email
----+----------------------
  1 | wallsingh@gmail.com
  2 | neha.verma@gmail.com
(2 rows)
```

كيف ينجز ذلك...

فيما يلي وصف للخطوات المذكورة في القسم السابق:

- في الخطوة ١ من القسم السابق، ننشئ أولا قاعدة بيانات Bucardo التي سوف تحتوي على معلومات حول برنامج Bucardo الخفي، وسوف تخلق أيضا مستخدم مميز بالاسم Bucardo .
- في الخطوة ٢، نقوم بإنشاء قواعد البيانات الخاصة بالمصدر والهدف من أجل النسخ، أي gamma1 و gamma2، على التوالي. أيضا ننشئ الجدول t1 على قاعدة البيانات gamma1 والذي سيستخدم للنسخ.
- في الخطوة ٣، نخبر Bucardo عن قواعد البيانات المصدر والهدف، وهذا هو gamma1 و gamma2، على التوالي التي سوف تشارك في النسخ
- في الخطوة ٤، ننشئ قطيع من قبل اسم myherd ويشمل الجدول t1 من قاعدة بيانات مصدر gamma1 التي ستكون جزءا من إعداد النسخ. يجب نسخ أية تغييرات تجرى على هذا الجدول من المصدر إلى قواعد البيانات الهدف.
- في الخطوات ٥ و ٦ من القسم السابق، ننشئ أساسا مزامنة المصدر و الهدف، والتي سوف تنسخ الجدول t1 في قطيع myherd ونسخها من قاعدة البيانات المصدر db1، وهذا هو gamma1، إلى قاعدة البيانات الهدف db2، هذا هو gamma2. مع إعداد تضبيط النسخ ، ثم نبدأ خدمة Bucardo في الخطوة ٧ من القسم السابق.
- نقوم باختبار إعداد النسخ في الخطوتين ٨ و ٩ من القسم السابق. في الخطوة ٨، نقوم بإدراج بعض السجلات في الجدول t1 على قاعدة البيانات gamma1 وفي الخطوة ٩، نسجل الدخول إلى قاعدة بيانات gamma2 والتحقق من ما إذا كانت السجلات المدرجة حديثا في الجدول t1 على قاعدة بيانات gamma1 قد نسخت عبر قاعدة بيانات gamma2. مجموعة نتائج الاستعلام SELECT من الجدول t1 في قاعدة بيانات gamma2 يؤكد أن السجلات المدرجة في قاعدة بيانات gamma1 نسخت بنجاح في قاعدة بيانات gamma2.

يمكنك الرجوع إلى الروابط التالية للحصول على معلومات حول النسخ Bucardo:

-http://blog.pscs.co.uk/postgresql-replication-and-bucardo/
-http://blog.endpoint.com/2014/06/bucardo-5-multimasterpostgres-released.html

النسخ باستخدام DRBD

في هذه الوصفة، سوف نغطي النسخ على مستوى الكتلة باستخدام DRBD لبوستجريسكل.

الاستعداد للعمل

يتطلب وجود حاسوب لينكس لتعمل هذا الإعداد. يتطلب هذا الإعداد واجهات الشبكة وعنوان عنقود IP. نفذت هذه الخطوات في جهاز سينتوس الإصدار ٦. وبعد أن غطينا إعداد بوستجريسكل في الفصول السابقة، فمن المفترض أن الحزم الضرورية والمتطلبات المسبقة مثبتة بالفعل.

سنستخدم الإعداد التالي في التسلسل الهرمي لدينا:

- يستخدم Node1.example.org عنوان IP الخاص بالشبكة المحلية ١٠.٠.٠.١٨١ ويستخدم ١٧٢.١٦.٠.١ لعمليات الانتقال
- يستخدم Node2.example.org عنوان IP على LAN في ١٠.٠.٠.١٨٢ وعنوان 172.16.0.2 IP لعمليات الانتقال: يستخدم dbip.example.org عنوان IP الخاص بالعنقود ١٠.٠.٠.١٨٠

كيف ينجز ذلك...

قم بتنفيذ التسلسل التالي من خطوات للحصول على النسخ على مستوى الكتلة باستخدام DRBD:

١. أولا، قم بتعطيل SELINUX مؤقتا، وتعيين SELINUX كمعطل، ثم قم بحفظ الملف:

```
vi    /etc/selinux/config
SELINUX=disabled
```

٢. في هذه الخطوة، غير اسم المضيف والبوابة لكلا العقد، وهي هنا واجهات الشبكة:

```
vi  /etc/sysconfig/network
#  For  node  1
NETWORKING=yes
NETWORKING_IPV6=no
HOSTNAME=node1.example.org
GATEWAY=10.0.0.2

#For  node  2
NETWORKING=yes
NETWORKING_IPV6=no
HOSTNAME=node2.example.org
GATEWAY=10.0.0.2
```

٣. في هذه الخطوة، نحن بحاجة إلى تضبيط واجهات الشبكة للعقدة الأولى، وهي node1:

- علينا أولا تضبيط قاعدة بيانات node1 الأولى:

```
vi  /etc/sysconfig/network-scripts/ifcfg-eth0
```

```
DEVICE=eth0
BOOTPROTO=static
IPADDR=10.0.0.181
NETMASK=255.255.255.0
ONBOOT=yes
HWADDR=a2:4e:7f:64:61:24
```

- ثم نقوم بتضبيط واجهة crossover/DRBD لـ node1

```
node1: vi /etc/sysconfig/network-scripts/ifcfg-eth1 DEVICE=eth1
BOOTPROTO=static
IPADDR=172.16.0.1
NETMASK=255.255.255.0
ONBOOT=yes
HWADDR=ee:df:ff:4a:5f:68
```

٤. في هذه الخطوة، نعد واجهات الشبكة للعقدة الثانية، وهذه هي، node2

```
vi /etc/sysconfig/network-scripts/ifcfg-eth0 DEVICE=eth0
BOOTPROTO=static
IPADDR=10.0.0.182
NETMASK=255.255.255.0
ONBOOT=yes
HWADDR=22:42:b1:5a:42:6f
```

- ثم نقوم بتضبيط واجهة crossover/DRBD للعقدة ٢

```
vi /etc/sysconfig/network-scripts/ifcfg-eth1 DEVICE=eth1
BOOTPROTO=static
IPADDR=172.16.0.2
NETMASK=255.255.255.0
ONBOOT=yes
HWADDR=6a:48:d2:70:26:5e
```

٥. في هذه الخطوة، سوف نعد DNS:

```
vi /etc/resolv.conf
search example.org
nameserver 10.0.0.2
```

- أيضا، نضبط قرار اسم المضيف الأساسي

```
vi /etc/hosts
127.0.0.1        localhost.localdomain          localhost
```

```
10.0.0.181      node1.example.org               node1
10.0.0.182      node2.example.org               node2
10.0.0.180      dbip.example.org                node2
```

٦. في هذه الخطوة، سوف نتحقق من شبكة الاتصال بين العقد:

- أولا، نقوم بإرسال إشارة ping إلى node2 من node1، أولا من خلال واجهة LAN ومن ثم عن طريق آيبي العبور:

```
root@node1 ~]# ping -c 2 node2
PING node2 (10.0.0.182) 56(84) bytes of data.
64 bytes from node2 (10.0.0.182): icmp_seq=1 ttl=64
time=0.089 ms
64 bytes from node2 (10.0.0.182): icmp_seq=2 ttl=64
time=0.082 ms
--- node2 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time
999ms
rtt min/avg/max/mdev = 0.082/0.085/0.089/0.009 ms
[root@node1 ~]# ping -c 2 172.16.0.2
PING 172.16.0.2 (172.16.0.2) 56(84) bytes of data.
64 bytes from 172.16.0.2: icmp_seq=1 ttl=64 time=0.083 ms
64 bytes from 172.16.0.2: icmp_seq=2 ttl=64 time=0.083 ms
--- 172.16.0.2 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time
999ms
rtt min/avg/max/mdev = 0.083/0.083/0.083/0.000 ms
```

- الآن، سنرسل إشارة ping إلى node1 من node2، أولا عبر واجهات LAN ومن ثم عن طريق IP العبور:

```
[root@node2 ~]# ping -c 2 node1
PING node1 (10.0.0.181) 56(84) bytes of data.
64 bytes from node1 (10.0.0.181): icmp_seq=1 ttl=64 time=0.068 ms
64 bytes from node1 (10.0.0.181): icmp_seq=2 ttl=64 time=0.063 ms
--- node1 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 999ms
rtt min/avg/max/mdev = 0.063/0.065/0.068/0.008 ms
```

- ثم نرسل إشارة ping عبر واجهة العبور:

```
[root@node2 ~]# ping -c 2 172.16.0.1
```

```
PING 172.16.0.1 (172.16.0.1) 56(84) bytes of data.
64 bytes from 172.16.0.1: icmp_seq=1 ttl=64 time=1.36 ms
64 bytes from 172.16.0.1: icmp_seq=2 ttl=64 time=0.075 ms
--- 172.16.0.1 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1001ms
rtt min/avg/max/mdev = 0.075/0.722/1.369/0.647 ms
```

٧. تثبيت الحزم اللازمة:

```
yum install -y  drbd83 kmod-drbd83
```

٨. في هذه الخطوة، نضبط DRBD على كلا العقد:

```
vi /etc/drbd.con
```

```
global {
    usage-count no;
}
common {
    syncer { rate 100M; }
    protocol      C;
}
resource postgres {
    startup {
       wfc-timeout 0;
       degr-wfc-timeout
       120;
    }
    disk { on-io-error detach; }
    on node1.example.org {
       device      /dev/drbd0;
       disk        /dev/sda5;
       address     172.16.0.1:7791;
       meta-disk   internal;
    }
    on node2.example.org {
       device      /dev/drbd0;
       disk        /dev/sda5;
       address     172.16.0.2:7791;
       meta-disk   internal;
    }
}
```

٩. بمجرد إعداد الملف drbd.conf لكلا العقدتين، نكتب البيانات الوصفية على الموارد postgres. وننفذ الخطوة التالية على كلا العقدتين:

```
[root@node1 ~]# drbdadm create-md postgres
Writing meta data...
initializing activity log
NOT initialized bitmap
New drbd meta data block successfully created.
root@node2 ~]# drbdadm create-md postgres
Writing meta data...
initializing activity log
NOT initialized bitmap
New drbd meta data block successfully created.
```

١٠. في هذه الخطوة، سوف نجلب الموارد. نفذ الأمر التالي على كلا العقدتين:

```
drbdadm up postgres
```

١١. في هذه الخطوة، يمكننا أن صنع المزامنة الأولية بين العقد. يمكن تنفيذ هذه الخطوة على العقدة الأساسية، وقد قمنا بتعيين node1 كعقدة أساسية:

```
drbdadm -- --overwrite-data-of-peer primary Postgres
```

١٢. لمراقبة التقدم المحرز في المزامنة وحالة الموارد DRBD، نلقي نظرة على ملف /proc/drbd:

```
[root@node1 ~]# cat /proc/drbd
version: 8.3.8 (api:88/proto:86-94)
GIT-hash: d78846e52224fd00562f7c225bcc25b2d422321d build by
mockbuild@builder10.centos.org, 2014-10-04 14:04:09
0: cs:SyncSource ro:Primary/Secondary ds:UpToDate/Inconsistent C
r----ns:48128 nr:0 dw:0 dr:48128 al:0 bm:2 lo:0 pe:0 ua:0 ap:0 ep:1
wo:b oos:8340188
[>....................] sync'ed: 0.6% (8144/8188)M delay_probe: 7
finish: 0:11:29 speed: 12,032 (12,032) K/sec
```

١٣. بمجرد اكتمال عملية المزامنة، يمكننا أن نلقي نظرة على كل من حالتي موارد postgres على كلتا العقدتين:

```
[root@node1 ~]# cat /proc/drbd
version: 8.3.8 (api:88/proto:86-94)
GIT-hash: d78846e52224fd00562f7c225bcc25b2d422321d build by
```

```
mockbuild@builder10.centos.org, 2014-10-04 14:04:09
0: cs:Connected ro:Primary/Secondary ds:UpToDate/UpToDate C r----
ns:8388316 nr:0 dw:0 dr:8388316 al:0 bm:512 lo:0 pe:0 ua:0 ap:0
ep:1 wo:b oos:0

[root@node2 ~]# cat /proc/drbd
version: 8.3.8 (api:88/proto:86-94)
GIT-hash: d78846e52224fd00562f7c225bcc25b2d422321d build by
mockbuild@builder10.centos.org, 2014-10-04 14:04:09
0: cs:Connected ro:Secondary/Primary ds:UpToDate/UpToDate C r----
ns:0 nr:8388316 dw:8388316 dr:0 al:0 bm:512 lo:0 pe:0 ua:0 ap:0
ep:1 wo:b oos:0
```

١٤. في هذه الخطوة، نحن على وشك بدء خدمات DRBD. على كلا العقدتين، أصدر الأمر التالي:

```
/etc/init.d/drbd start
```

١٥. من أجل تضبيط دليل البيانات وإعداده باستخدام DRBD، سيتعين علينا تنسيق جهاز DRBD وتركيبه. بعد ذلك، نقوم بتضبيط دليل البيانات:

- قم بإصدار الأوامر التالية على node1:

```
mkfs.ext4 /dev/drbd0
mount -t ext4 /dev/drbd0 /var/lib/pgsql/9.3
chown postgres.postgres /var/lib/pgsql/9.3
```

- بعد ذلك، قم بتسجيل الدخول كمستخدم postgres على node1 وقم بتضبيط قاعدة البيانات:

```
su – postgres
 initdb /var/lib/pgsql/9.3/data
exit
```

١٦. في هذه الخطوة، يمكننا تفعيل المصادقة الموثوق بها، وسوف نقوم بتضبيط المعلمات المطلوبة لإعداد بوستجريسكل في ملف postgresql.conf.

- على node1، نفذ الخطوات التالية:

```
echo "host all all 10.0.0.181/32 trust" >> /var/lib/
pgsql/9.3/data/pg_hba.conf
echo "host all all 10.0.0.182/32 trust" >> /var/lib/
pgsql/9.3/data/pg_hba.conf
echo "host all all 10.0.0.180/32 trust" >> /var/lib/
```

```
pgsql/9.3/data/pg_hba.conf
```

- ثم نقوم بتضبيط المعلمات الضرورية في الملف postgresql.conf.

```
vi /var/lib/pgsql/9.3/data/postgresql.conf
listen_addresses = '*'
```

١٧. بمجرد تغيير المعلمات المذكورة سابقا في ملف postgresql. conf ، فإن الخطوة التالية ستكون بدء بوستجريسكل. عن طريق تنفيذ الأمر التالي على node1:

```
service postgresql-9.3 start
```

١٨. سنقوم بعد ذلك بإنشاء مستخدم مسؤول لإدارة بوستجريسكل. على node1، نفذ الأمر التالي وعندما يطلب منك كلمة مرور، يمكنك اختيار أي واحدة. ومع ذلك، من أجل توضيح هذه العملية، سنستخدم الكلمة الرئيسية للمشرف نفسها ككلمة مرور:

```
su — postgres
createdb test
pgbench -test
pgbench -i test
psql -U admin -d test
```

١٩. وفي هذه الخطوة، سننشئ قاعدة بيانات ونملأها بالبيانات. على node1، قم بتنفيذ الخطوات التالية ثم قم بالنفاذ إلى قاعدة البيانات:

```
su - postgres
createdb test
pgbench -test
pgbench -i test
psql -U admin -d test
test=# select * from pgbench_tellers;
tid | bid | tbalance | filler
-----+-----+----------+--------
  1 |   1 |        0 |
  2 |   1 |        0 |
  3 |   1 |        0 |
  4 |   1 |        0 |
  5 |   1 |        0 |
  6 |   1 |        0 |
  7 |   1 |        0 |
  8 |   1 |        0 |
  9 |   1 |        0 |
 10 |   1 |        0 |
```

```
(10 registros)
```

٢٠. في هذه الخطوة، سوف نقوم باختبار نسخ مستوى كتلة ونرى ما إذا كان بوستجريسكل يعمل في node2. على عقدة ١، نفذ الأوامر التالية:

- سنوقف أولا بوستجريسكل على العقدة ١:

```
service postgresql-9.3 stop
```

- ثم، سنقوم بإلغاء تحميل الجهاز DRBDعلى العقدة ١:

```
umount /dev/drbd0
```

- الآن، سوف نقوم بإعداد node1 كعقدة ثانوية:

```
drbdadm secondary postgres
```

- بعد ذلك، سنقوم بتضبيط node2 كعقدة أساسية:

```
drbdadm primary postgres
```

- في هذه الخطوة، قم بتثبيت جهاز DRBD:

```
mount -t ext3 /dev/drbd0 /var/lib/pgsql/9.3
```

- ثم نبدأ خدمة بوستجريسكل على node2:

```
service postgresql-9.3 start
```

- الآن، سوف نرى ما إذا كنا قادرين على النفاذ إلى قاعدة بيانات الاختبار على node2:

```
psql -u admin -d test
test=# select * from pgbench_tellers;
tid | bid | tbalance | filler
-----+-----+----------+--------
  1 |   1 |        0 |
  2 |   1 |        0 |
  3 |   1 |        0 |
  4 |   1 |        0 |
  5 |   1 |        0 |
  6 |   1 |        0 |
  7 |   1 |        0 |
  8 |   1 |        0 |
  9 |   1 |        0 |
 10 |   1 |        0 |
```

```
(10 registros)
```

كيف ينجز ذلك...

في الخطوات الأولية، من الخطوات ١-٦، قمنا بضبط العقد، وهي node1 و node2، ثم إعداد اتصالات الشبكة، وتضبيط DNS. في الخطوة ٦، قمنا باختبار اتصالات الشبكة بين العقدة ١ والعقدة ٢ على واجهة LAN وكذلك على واجهة العبور crossover. وقد تلقينا رسائل استجابة صدى ناجحة بعد إجراء اختبارات طلب ping، وهذا يدل على أن اتصال الشبكة تم تضبيطه بنجاح.

في الخطوة ٨، أنشأنا الملف drbd.conf في كلتا العقدتين. وهنا مقتطف من الملف drbd.conf:

```
global {
    usage-count no;
}
common {
    syncer { rate 100M; }
    protocol      C;
}
resource postgres {
    startup {
       wfc-timeout 0;
       degr-wfc-timeout
       120;
    }
    disk { on-io-error detach; }
    on node1.example.org {
       device      /dev/drbd0;
       disk        /dev/sdb;
       address     172.16.0.1:7791;
       meta-disk   internal;
    }
    on node2.example.org {
       device      /dev/drbd0;
       disk        /dev/sdb;
       address     172.16.0.2:7791;
       meta-disk   internal;
    }
}
```

وببساطة، باستخدام التضبيط المذكور سابقا، نحن نعد مورد postgres ونكون واجهة DRBD و /dev/drbd0، الذي أعد على العقدتين، عقدة ١ والعقدة ٢. هذا هو أساسا ما يسبب أن يكون النسخ على مستوى الكتلة ناجحا. في الخطوة ١١ من القسم السابق، يمكنك أن ترى أننا قمنا في البداية بإعداد العقدة ١ كعقدة أساسية والعقدة ٢ بمثابة العقدة الثانوية في هذه المرحلة. ثم أعددنا بوستجريسكل على العقدة ١ من الخطوة ١٥ فصاعدا. ومن الخطوة ٢٠ فصاعدا، أجرينا اختبار تجاوز الفشل. فقمنا أولا بتعيين العقدة ١ كعقدة ثانوية، ثم ألغينا تحميل نظام الملفات، ثم قمنا بإعداد العقدة ٢ كعقدة أساسية؛ ثم ربط نظام الملفات و تشغيل خادم بوستجريسكل. بعد ذلك، قمنا باختبار رؤية السجلات في العقدة ٢، أنشئت قاعدة البيانات الاختبارية في الخطوة ١٩ من الخطوات السابقة، صارت قابلة للوصول في العقدة ٢ كذلك الجداول في مخطط pgbench في الخطوة ٢٠. وهكذا، قام DRBD بالنسخ على مستوى الكتلة، وإذا كانت إحدى العقد غير متوفرة، يمكننا ضبط والاستمرار في تشغيل بوستجريسكل في العقدة الثانوية حيث ستقوم بأخذ دور الخادم الرئيسي.

إعداد عنقود postgres-XC

في هذه الوصفة، سنقوم بإعداد عنقود postgres-XC.

الاستعداد للعمل

هنا، نحن بحاجة إلى تثبيت وإعداد postgres-XC. نفذت هذه الخطوات على جهاز لينكس سينتوس الإصدار ٦.

نفذ مجموعة الخطوات التالية:

١. أولا، اذهب إلى http://sourceforge.net/projects/postgres-xc/ من أجل تحميل برنامج postgres-XC

٢. في هذه الخطوة، قم باستخراج ملفات tarball والذهاب إلى الدليل الذي أنشيء حديثا:

```
tar -zxvf pgxc-v1.0.4.tar.gz
cd pgxc-v1.0.4
```

٤. قبل إنشاء البرنامج وتجميعه، ستكون الخطوة التالية هي تثبيت حزم المتطلبات الرئيسية التالية:

```
yum -y install readline*
yum -y install bison*
yum -y install flex*
```

٥. الآن، نحن ذاهبون لبناء وتجميع البرنامج. سنحدد أيضا موقعا لاستخدامه كالبادئة:

```
mkdir -p /opt/Postgres-xc
```

```
chown -R postgres:postgres /opt/Postgres-xc/
./configure --prefix=/opt/Postgres-xc/
make
make install
```

كيف ينجز ذلك...

الآن، مع اكتمال التثبيت، نحن بحاجة إلى ضبط إعدادات postgres-XC.

قم بتنفيذ الخطوات التالية:

١. سنقوم الآن بإعداد GTM (مدير المعاملات العالمية). لهذا الغرض، سنقوم أولا بإنشاء دليل لمدير المعاملات العالمية، وتعيين الأذونات، ثم تهيئة مدير المعاملات العالمية:

```
mkdir -p /usr/local/pgsql/data_gtm
chmod -R 700 /usr/local/pgsql/data_gtm
/opt/Postgres-xc/bin/initgtm -Z gtm -D /usr/local/pgsql/data_gtm
```

٢. سنقوم الآن بتضبيط المعلمات في ملف gtm.conf، الذي أنشئ كجزء من الخطوة السابقة عندما قمنا بتهيئة مدير المعاملات العالمية، وبدء مدير المعاملات العالمية:

```
nodename = 'GTM_Node'
listen_addresses = '*'
port = 7777
```

بمجرد أن غيرت هذه المعلمات، يمكننا بعد ذلك إعداد مدير المعاملات العالمية:

```
/opt/Postgres-xc/bin/gtm_ctl -Z gtm start -D /opt/Postgres-xc/
data_gtm
Server Started
```

٣. مع إعداد مدير المعاملات العالمية وتشغيله، سنقوم بإعداد عقدة المنسق. لهذا الغرض، سنقوم أولا بإنشاء دليل للمنسق، وتعيين الأذونات، ومن ثم تضبيط المنسق:

```
mkdir -p /opt/Postgres-xc/data_coord1
chmod  -R 700 /opt/Postgres-xc/data_coord1
/opt/Postgres-xc/bin/pg_ctl -D /opt/Postgres-xc/data_coord1/ -o '--nodename
coord1' initdb
```

٤. في الخطوة التالية، سنقوم بتضبيط المعلمات اللازمة في ملف postgresql.conf. بحيث يعد بطريقة حيث تستخدم المنسق كعقدة اتصال لمدير المعاملات العالمية. أيضا، بمجرد ضبط المعلمات اللازمة، سوف نشغل المنسق:

```
listen_addresses = '*'
port = 2345
gtm_host = 'localhost'
gtm_port = 7777
pgxc_node_name = 'coord1'
pooler_port = 2344
min_pool_size = 1
max_pool_size = 100
persistent_datanode_
connections = on
max_coordinators = 16
max_datanodes = 16
```

عندما تضبط المعلمات اللازمة، سوف نبدأ المنسق كما يلي:

```
/opt/Postgres-xc/bin/pg_ctl start -D /opt/Postgres-xc/data_coord1/
-Z coordinator -l /tmp/coord
```

٥. سنقوم الآن بإعداد عقدة البيانات الأولى. لهذا الغرض، سنقوم بإنشاء دليل، وتعيين الأذونات الخاصة به، ثم تهيئته ذلك:

```
mkdir -p /opt/Postgres-xc/data_node1
chmod -R 700 /opt/Postgres-xc/data_node1

/opt/Postgres-xc/bin/pg_ctl -D /opt/Postgres-xc/data_node1/ -o
'--nodename datanode1' initdb
```

٦. في الخطوة التالية، سنقوم بتضبيط المعلمات اللازمة لعقدة البيانات الأولى ثم نقوم بتشغيل عقدة البيانات. نقوم بإجراء هذه التغييرات في ملف postgresql.conf:

```
vi postgresql.conf
listen_addresses = '*'
port = 1234
gtm_host = 'localhost'
gtm_port = 7777
pgxc_node_name = 'datanode1'
```

وبمجرد إجراء هذه التغييرات، يمكننا إطلاق عقدة البيانات الأولى:

```
/opt/Postgres-xc/bin/pg_ctl start -D /opt/Postgres-xc/data_node1 -Z datanode
-l /tmp/datanode1_log
```

٧. في هذه الخطوة، سوف نقوم بإعداد عقدة البيانات الثانية. لهذا الغرض، سوف نقوم بتضبيط دليل عقدة البيانات الثانية، وتعيين أذونات، ثم تضبيط ذلك:

```
mkdir -p /opt/Postgres-xc/data_node2/
chmod -R 700 /opt/Postgres-xc/data_node2/
/opt/Postgres-xc/bin/pg_ctl -D /opt/Postgres-xc/data_node2/ -o
'--nodename datanode2' initdb
```

٨. في هذه الخطوة، سنقوم بتضبيط المعلمات لعقدة البيانات الثانية، وبمجرد الانتهاء من ذلك، سنبدأ عقدة البيانات الثانية:

```
vi postgresql.conf
listen_addresses = '*'
port = 1233
gtm_host = 'localhost'
gtm_port = 7777
pgxc_node_name = 'datanode2'
```

وبمجرد إجراء هذه التغييرات المعلمة اللازمة، سنبدأ عقدة البيانات الثانية:

```
/opt/Postgres-xc/bin/pg_ctl start -D /opt/Postgres-xc/data_node2
-Z datanode -l /tmp/datanode2_log
```

٩. في هذه الخطوة، سوف نسجل العقد البيانات الأولى والثانية على عقدة المنسق:

```
cd /opt/Postgres-xc/bin/

postgres=# CREATE NODE datanode1 WITH ( TYPE = DATANODE , HOST = LOCALHOST , PORT
= 1234 );
CREATE NODE
postgres=# CREATE NODE datanode2 WITH ( TYPE = DATANODE , HOST = LOCALHOST , PORT
= 1233 );
CREATE NODE
```

١٠. الآن، مع اكتمال إعداد Postgres-XC ، سنبدأ في توزيع البيانات عن طريق النسخ:

```
psql -p 2345
postgres=# CREATE TABLE DIST (T INT) DISTRIBUTE BY REPLICATION TO NODE
datanode1,datanode2;
```

```
CREATE TABLE
postgres=#INSERT INTO DIST SELECT * FROM generate_series(1, 100);
INSERT 0 100
postgres=# EXPLAIN ANALYZE SELECT * FROM DIST;
                              QUERY PLAN
---------------------------------------------------------------------------
 Data Node Scan on "__REMOTE_FQS_QUERY__"  (cost=0.00..0.00 rows=0 width=0) (act
ual time=0.880..1.010 rows=100 loops=1)
   Node/s: datanode1
 Total runtime: 1.076 ms
(3 rows)
```

سوف نقوم الآن بتسجيل الدخول إلى datanode1 و datanode2، لمعرفة ما إذا نسخت هذه السجلات على جدول DIST:

```
psql -p 1234
postgres=# select count(*) from DIST;
 count
-------
   100
(1 row)
psql -p 1233
postgres=# select count(*) from DIST;
 count
-------
   100
(1 row)
```

١١. الآن، سنقوم باختبار التوزيع عن طريق الهاش:

سجل الدخول إلى عقدة المنسق:

```
psql -p 2345
CREATE TABLE t_test (id int4) DISTRIBUTE BY HASH (id);
INSERT INTO t_test SELECT * FROM generate_series(1, 1000);
```

سنقوم الآن بتسجيل الدخول إلى datanode1 و datanode2 ونرى كم عدد السجلات التي نسخت هناك:

```
psql -p 1233
postgres=# select count(*) from t_test;
 count
-------
   508
(1 row)
psql -p 1234
```

```
postgres=# select count(*) from t_test;
 count
-------
   492
(1 row)
```

كيف تعمل...

في معمارية postgres-XC بأكملها، استخدمنا الإعداد التالي. نحن نستخدم مدير المعاملات العالمية، ومنسق، واثنين من العقد البيانات. وسوف نناقش وظائف كل واحد منهم:

- GTM : مدير المعاملات العالمية يستخدم لتوفير وجهة نظر متسقة من البيانات. وتقدم وجهة نظر متسقة أساسا من خلال لقطة على نطاق العنقود. مدير المعاملات العالمية مسؤول أيضا عن إنشاء معرفات المعاملات العالمية، والتي هي ضرورية لأن المعاملات تحتاج إلى تنسيق على مستوى العنقود.
- المنسق Coordinator: يخدم هذا كنقطة إدخال للتطبيقات ويستخدم من قبلها. المنسق يعتبر مسؤول عن تحليل SQL، وإنشاء خطة تنفيذ عامة، و تنفيذ SQL عامة.
- عقدة البيانات Data node: تستخدم عقدة البيانات لحفظ البيانات لعنقود بوستجريسكل. تحفظ عقدة بيانات واحدة أو أكثر كل أو جزء من البيانات داخل العنقود.

سنناقش الآن الخطوات المختلفة التي قمنا بها في القسم السابق:

- هنا، سوف نناقش الخطوات ١ و ٢ من القسم السابق. هذا الإعداد هو كل شيء عن تضبيط مدير المعاملات العالمية. مبدئيا نقوم بضبط دليل لمدير المعاملات العالمية، ثم تعيين الأذونات، ثم تضبيط الدليل ثم تشغيل مدير المعاملات العالمية في وقت لاحق.
- الخطوات ٣ و ٤ من القسم السابق كلها عن تضبيط عقدة المنسق. نبدأ في تضبيط دليل للمنسق، ثم تعيين أذونات، ثم تهيئة الدليل ثم تشغيل المنسق.
- بعد ذلك، سنناقش الخطوتين ٥ و ٦ من القسم السابق. هذا الإعداد هو كل شيء عن تضبيط عقدة البيانات الأولى. مبدئيا تضبيط دليل ل datanode1، ثم تعيين الأذونات، ثم تهيئة الدليل ثم تشغيل عقدة البيانات الأولى.
- تناقش الخطوات ٧ و ٨ من القسم السابق تضبيط عقدة البيانات الثانية. نقوم في البداية بتضبيط دليل لـ datanode2، ثم تعيين الأذونات، ثم تهيئة الدليل ثم تشغيل عقدة البيانات الثانية.

- في الخطوة ٩، نقوم أولا بتسجيل الدخول إلى عقدة المنسق، ثم نسجل العقد datanode1 و datanode2 مع عقدة المنسق.
- في الخطوات ١٠ و ١١، نحن أساسا نختبر عنقود Postgres-XC. في الخطوة ١٠، نقوم بتسجيل الدخول إلى عقدة المنسق وننشئ جدول DIST ثم نوزع هذا الجدول بواسطة النسخ إلى datanode1 و datanode2. ثم نقوم بإنشاء سلسلة وإدراج حوالي ١٠٠ سجل في الجدول DIST. التوزيع بواسطة النسخ إلى عقد البيانات يعني أنه يجب نسخ البيانات في كل من العقد لجدول DIST. ثم نقوم بتسجيل الدخول إلى عقدة datanode1 ثم حساب عدد السجلات من الجدول DIST؛ العدد هو ١٠٠. نحصل على نفس الملاحظة لجدول DIST عند تسجيل الدخول إلى عقدة datanode2. ويظهر ذلك بشكل فعال في الخطوة ١٠ من القسم السابق؛ وبالتالي، نسخت كافة سجلات الجدول DIST عبر كل عقد البيانات. في الخطوة ١١، نقوم بتسجيل الدخول إلى عقدة المنسق وإنشاء جدول t_test؛ ثم نقوم بتوزيع الجدول بواسطة التجزئة وإدراج ١٠٠٠ سجل في الجدول. ثم نقوم بتسجيل الدخول إلى عقدة البيانات الأولى، ويمكننا أن نرى ٥٠٨ سجلات هنا. ثم نقوم بتسجيل الدخول إلى عقدة البيانات الثانية، ويمكننا أن نرى ٤٩٢ سجلات في الجدول t_test في عقدة datanode2. ما نراه هو التوزيع المتساوي لتخزين سجلات الجدول t_test بين عقد البيانات ١ و ٢.

٨- تجميع الاتصالات

في هذا الفصل، سنغطي الوصفات التالية:

- تثبيت pgpool
- ضبط pgpool واختبار الإعداد
- بدء و إيقاف pgpool
- إعداد pgbouncer
- تجميع الاتصالات باستخدام pgbouncer
- إدارة pgbouncer

مقدمة

الأداة المساعدة pgpool-II هي في الأساس حل الوسيط تعمل كواجهة بين خادم بوستجريسكل وتطبيق عميل بوستجريسكل. الأداة pgpool-II هي بمثابة بروكسي بين خلفية بوستجريسكل وبروتوكولات الواجهة وهي تربط الاتصالات بين الاثنين. تقوم الأداة pgpool-II بعمل نسخة مخبأة من الاتصالات بخوادم بوستجريسكل وتعيد استخدامها كلما كان هنالك اتصال جديد بنفس الخصائص ، وبالتالي تقلل كلفة تفاوض الاتصال من مثل الاستيثاق والتشفير، وتحسين الإنتاجية الإجمالية.

في الواقع، الأداة المساعدة pgpool-II توفر الكثير من الميزات وليس مجرد تجميع الاتصالات. حيث توفر موازنة التحميل وأنماط النسخ جنبا إلى جنب مع ميزة الاستعلام المتوازي.

الأداة المساعدة **pgbouncer** هي مجمعة اتصالات خفيفة الوزن لبوستجريسكل. تتصل التطبيقات بمنفذ pgbouncer تماما مثل اتصالها بقاعدة بيانات بوستجريسكل على منفذ قاعدة البيانات. باستخدام الأداة المساعدة pgbouncer، يمكننا خفض تأثير الاتصال الزائد على خادم بوستجريسكل. توفر الأداة pgbouncer تجميع الاتصال عن طريق إعادة استخدام الاتصالات الموجودة.

الفرق بين pgbouncer و pgpool-II هو أن pgbouncer خفيفة الوزن ومخصصة لغرض تجميع الاتصال، في حين تقدم pgpool-II المزيد من الميزات مثل النسخ replication، وموازنة الحمل، وميزة الاستعلام المتوازي بالإضافة إلى تجميع الاتصال.

في هذا الفصل، سوف نشير إلى pgpool-II كـ pgpool لغرض البساطة.

تثبيت pgpool

هنا نجد وصفة تثبيت pgpool وضبطها.

الاستعداد للعمل

تثبيت pgpool من المصدر يتطلب gcc 2.9 أو أحدث و GNU make. وبما أن pgpool يرتبط بمكتبة libpq ، فإن مكتبة libpq وترويسات تطويرها، يجب أن تكون مثبتة قبل تثبيت pgpool. أيضا، يجب أن تكون مكتبة OpenSSL مثبتة من أجل تفعيل دعم OpenSSL في pgpool.

إذا كنت تبني من المصدر، فاتبع الخطوات التالية:

١. قم بتحميل أحدث tarball من pgpool من الموقع التالي:

http://www.pgpool.net/mediawiki/index.php/Downloads

٢. الخطوة التالية هي استخراج tarball pgpool ودخول دليل المصدر:

```
tar -xzf pgpool-II-3.4.0.tar.gz
cd pgpool-II-3.4.0
```

٣. قم ببناء وتجميع وتركيب برنامج pgpool:

```
./configure –prefix=/usr/local --sysconfdir=/etc/pgpool/
make
make install
```

كيف ينجز ذلك...

لتثبيت pgpool في توزيعة تستند إلى دبيان أو أوبنتو ، يمكننا تنفيذ هذا الأمر:

```
apt-get install pgpool2
```

على ريد هات، أو فيدورا Fedora، أوسينتوس CentOS، أو أي توزيعات لينكس أخرى مستندة إلى RHEL تستخدم الأمر التالي. وتجدر الإشارة إلى أن اسم الحزمة المستخدمة هو ما هو يوجد الآن لـ pgpool، فالكلمة ٩٣ الكلمة المستخدمة في نهاية تتعلق بالإصدارة المصغرة من بوستجريسكل. قد يتغير لاحقا عندما تصدر تحديثات جديدة:

```
yum install pgpool-II-93
```

الخطوات التالية تصبح قابلة للتطبيق عند استخدام مدير حزم نظام التشغيل مثل yum لتثبيت pgpool، وعند التحميل والتجميع من المصدر. في الأساس، في الخطوات التالية، نحن نقوم بانشاء دليل لـ pgpool حيث أنه يمكن الحفاظ على سجلات النشاط وملفات قفل الخدمة الخاصة بها:

١. في هذه الخطوة، سوف نقوم بإنشاء موقع حيث يمكن لـ pgpool الحفاظ على سجلات النشاط:

```
mkdir /var/log/pgpool
chown —R postgres:postgres /var/log/pgpool
```

٢. الخطوة التالية هي إنشاء دليل حيث يمكن لـ pgpool تخزين ملفات قفل الخدمة:

```
mkdir /var/run/pgpool
chown —R postgres:postgres /var/run/pgpool
```

كيف تعمل...

إذا كنت تستخدم مدير حزم نظام تشغيل معين لتثبيت pgpool، فإن ملفات الضبط وملفات السجلات المطلوبة ستنشأ تلقائيا. ومع ذلك، إذا كنت في صدد تثبيت pgpool القائم على المصدر بالكامل، فإن هناك بعض الخطوات الإضافية المطلوبة. الخطوة الأولى هي تشغيل سكربت الإعداد ثم بناء وتجميع pgpool. بعد تثبيت pgpool، ستكون بحاجة لإنشاء المجلدات حيث يمكن لـ pgpool الحفاظ على سجلات النشاط وملفات قفل الخدمة. كل هذه الخطوات تحتاج إلى أن تنجز يدويا كما هو مبين في الخطوات ١، ٢، و ٣، على التوالي، في قسم الاستعداد للعمل.

ضبط pgpool واختبار الإعداد

في هذه الوصفة، نحن في طريقنا لضبط pgpool وتبيين كيفية إجراء اتصالات.

الاستعداد للعمل

قبل تشغيل pgpool، إذا كنت تقوم بتحميل مصدر tarball، فإن برنامج pgpool يحتاج إلى بناء وتجميع. تظهر هذه الخطوات في الوصفة الأولى من هذا الفصل.

أيضا، سوف نختبر النسخ باستخدام pgpool. لهذا الغرض، قمنا بإعداد مجلدين للبيانات على نفس الخادم. وسوف سيعملان بمثابة عقدتين.

على افتراض أن دليل البيانات الافتراضي، /var/lib/pgsql/9.3/data، والذي سيكون بمثابة العقدة ٠، قد أعد بالفعل، سنقوم الآن بإعداد دليل بيانات آخر الذي سيكون بمثابة العقدة ١:

```
initdb -D /var/lib/pgsql/9.3/data1
```

بمجرد إعداد مجلد البيانات، الخطوة التالية هي تغيير رقم المنفذ لمجلد البيانات الجديد. و ذلك لأن لمجلدي البيانات لا يمكن أن يكون لهما نفس رقم المنفذ. وبما أن رقم المنفذ ٥٤٣٢ قد استخدم بالفعل لمجلد البيانات الأول، سنقوم بتعيين رقم المنفذ ٥٤٣٣ في ملف postgresql.conf لمجلد البيانات الجديد ثم قم بتشغيل الخادم باستخدام هذا الإعداد:

```
cd /var/lib/pgsql/9.3/data1
vi postgresql.conf
port=5433
```

بمجرد حفظ الملف، قم ببدء تشغيل الخادم الجديد:

```
pg_ctl -D /var/lib/pgsql/9.3/data1 start
```

كيف ينجز ذلك...

سوف نتبع هذا التسلسل من الخطوات لضبط pgpool وتشغيل التثبيت:

١. بعد تثبيت pgpool كما هو موضح في الوصفة الأولى من الفصل، فإن الخطوة التالية ستكون نسخ ملفات الضبط من دليل العينة مع الإعدادات الافتراضية. ستعدل لاحقا وفقا لمتطلباتنا:

```
cd /etc/pgpool-II-93
cp pgpool.conf.sample /etc/pgpool.conf
cp pcp.conf.sample /etc/pcp.conf
```

٢. الخطوة التالية هي تحديد اسم المستخدم وكلمة المرور في ملف pcp.conf، وهو ملف استيثاق لـ pgpool. في الأساس، من أجل استخدام أوامر PCP، فإنه يتطلب استيثاق المستخدم. هذه الآلية تختلف عن استيثاق مستخدم بوستجريسكل. تشفر كلمات المرور في هيئة MD5. للحصول على تجزئة MD5 للمستخدم، علينا

استخدام الأداة pg_md5 كما هو موضح في الأمر التالي. بمجرد توليد تجزئة MD5، يمكن استخدامها لتخزين كلمة مرور MD5 في ملف pcp.conf:

```
pg_md5 postgres
e8a48653851e28c69d0506508fb27fc5
vi /etc/pcp.conf
postgres:e8a48653851e28c69d0506508fb27fc5
```

٣. الآن نقوم بتحرير ملف التضبيط pgpool.conf لضبط إعدادات pgpool الخاص بك:

```
listen_addresses = 'localhost'
port = 9999
socket_dir = '/tmp'
pcp_port = 9898
pcp_socket_dir = '/tmp'
backend_hostname0 = 'localhost'
backend_port0 = 5432
backend_weight0 = 1
backend_data_directory0 = '/var/lig/pgsql/9.3/data'
backend_flag0 = 'ALLOW_TO_FAILOVER'
backend_hostname1 = 'localhost'
backend_port1 = 5433
backend_weight1 = 1
backend_data_directory1 = '/var/lib/pgsql/9.3/data1'
backend_flag1 = 'ALLOW_TO_FAILOVER'
enable_pool_hba = off
pool_passwd = 'pool_passwd'
authentication_timeout = 60
ssl = off
num_init_children = 32

max_pool = 4
child_life_time = 300
child_max_connections = 0
connection_life_time = 0
client_idle_limit = 0
connection_cache = on
reset_query_list = 'ABORT; DISCARD ALL'
replication_mode = on
master_slave_mode = off
replicate_select = off
insert_lock = on
load_balance_mode = on
```

```
ignore_leading_white_space = on
white_function_list = ''
black_function_list = 'nextval,setval'
```

٤. بمجرد ضبط المعلمات السابقة وحفظها في ملف pgpool.conf، فإن الخطوة التالية هي تشغيل pgpool والبدء في قبول الاتصالات إلى عنقود بوستجريسكل باستخدام pgpool:

```
pgpool -f /etc/pgpool.conf -F /etc/pcp.conf
psql -p 9999 postgres postgres
```

٥. الآن وقد بدأت pgpool، ينبغي أن نرى مجموعة من العمليات:

```
-bash-4.1$ ps ax |grep pool
28778 ? Ss 0:00 pgpool -f /etc/pgpool.conf -F /etc/pcp.
conf
28779 ? S 0:00 pgpool: wait for connection request
28780 ? S 0:00 pgpool: wait for connection request

28781 ? S 0:00 pgpool: wait for connection request
28782 ? S 0:00 pgpool: wait for connection request
28783 ? S 0:00 pgpool: wait for connection request
28784 ? S 0:00 pgpool: wait for connection request
28785 ? S 0:00 pgpool: wait for connection request
28786 ? S 0:00 pgpool: wait for connection request
28787 ? S 0:00 pgpool: wait for connection request
28788 ? S 0:00 pgpool: wait for connection request
28789 ? S 0:00 pgpool: wait for connection request
28790 ? S 0:00 pgpool: wait for connection request
28811 ? S 0:00 pgpool: PCP: wait for connection
request
28812 ? S 0:00 pgpool: worker process
28849 pts/2 S+ 0:00 grep pool
```

٦. قبل الاتصال ب pgpool وبدء تنفيذ الاستعلامات، يجب علينا التحقق من حالة العقد المشاركة في العنقود. لهذا الغرض سوف نستخدم أداة تسمى pcp_node_info. ونظرا لأننا نستخدم نفس الخادم للتثبيت, فإن العقدة ٠ والعقدة ١ هي مجلدات بيانات موجودة في /var/lib/pgsql/9.3/data و /var/lib/pgsql/9.3/data1:

```
-bash-4.1$ pcp_node_info 5 localhost 9898 postgres postgres 0
localhost 5432 1 0.500000
-bash-4.1$ pcp_node_info 5 localhost 9898 postgres postgres 1
localhost 5433 1 0.500000
```

٧. الآن بعد أن كانت العقدتين مشاركتين في الكتلة، فإن الخطوة التالية هي الاتصال ب pgpool، وإنشاء جدول، وإدراج بعض السجلات في هذا الجدول:

```
psql -p 9999
postgres=# create table emp(age int);
CREATE TABLE
postgres=# insert into emp values (1);
INSERT 0 1
postgres=# insert into emp values (2);
INSERT 0 1
postgres=# insert into emp values (3);
INSERT 0 1
postgres=# insert into emp values (4);
INSERT 0 1
postgres=# insert into emp values (5);
INSERT 0 1
postgres=# insert into emp values (6);
INSERT 0 1
postgres=# \dt
        List of relations
 Schema | Name | Type  |  Owner
--------+------+-------+----------
 public | emp  | table | postgres
(1 row)
postgres=# select * from emp;
 age
-----
   1
   2
   3
   4
   5
   6
(6 rows)
```

٨. الآن سوف نقوم باختبار النسخ من خلال الاتصال بالمنافذ ٥٤٣٢ و ٥٤٣٣، ونرى الجدول والسجلات المناظرة التي أدرجت فيه أثناء الاتصال بـ pgpool:

```
psql -p 5433
postgres=# \dt
        List of relations
 Schema | Name | Type  |  Owner
--------+------+-------+----------
 public | emp  | table | postgres
(1 row)
```

```
postgres=# select * from emp;
 age
-----
   1
   2
   3
   4
   5
   6
(6 rows)
-bash-4.1$ psql -p 5432
postgres=# \dt
        List of relations
 Schema | Name | Type  |  Owner
--------+------+-------+----------
 public | emp  | table | postgres
(1 row)
postgres=# select * from emp;
 age
-----
   1
   2
   3
   4
   5
   6
(6 rows)
```

كيف تعمل...

لنناقش الآن بعض المعلمات التي تمت ضبطها في القسم السابق:

- listen_addresses: نقوم بضبط list_addresses إلى * لأننا نريد الاستماع إلى جميع عناوين IP وليس عنوان IP معين.
- port: هذا يحدد منفذ pgpool الذي سيستمع إليه النظام عند قبول اتصالات قاعدة البيانات.
- backend_hostname0: يشير هذا إلى اسم المضيف لقاعدة البيانات الأولى في الإعداد لدينا.

وبالمثل، أنشأنا backend_hostname1 للعقدة الثانية.

- backend_port0: هذا هو منفذ بروتوكول التحكم في نقل البيانات TCP للنظام، أي النظام الذي حدد من قبل في قيمة backend_hostname0 التي حيث توجد قاعدة البيانات. وبالمثل، وضعنا backend_port1 للعقدة الثانية.

- backend_weight0: هذا هو الوزن المخصص للعقدة التي حددها اسم المضيف الذي حصل عليه من backend_hostname0. أساسا في pgpool، تعين الأوزان إلى العقد الفردية. وترسل المزيد من الطلبات إلى العقدة ذات قيمة وزن أعلى. وبالمثل، أنشأنا backend_weight1 للعقدة الثانية.
- backend_data_directory0: يمثل هذا مجلد البيانات، أي PGDATA للمضيف الذي حدد بواسطة قيمة backend_hostname0. وبالمثل، قمنا بإعداد قيمة backend_data_directory1 للعقدة الثانية.
- connection_cache: لتفعيل وضع تجمع الاتصال، نحن بحاجة إلى تشغيل connection_cache
- max_pool: تحدد هذه القيمة الحد الأقصى لحجم التجمع لكل طفل (فرع). عدد الاتصالات من pgpool إلى قواعد البيانات التي خلف pgpool قد تصل إلى الحد في max_pool * num_init_children.

في حالتنا، قمنا بضبط max_pool إلى ٤ و num_init_children إلى ٣٢، وهما القيمة الافتراضية. لذلك عندما تضاعف، قد يصل إجمالي عدد الاتصالات من pgpool إلى الخلفية بحد أقصى ١٢٨. تذكر أن قيمة max_pool * num_init_children يجب أن تكون دائما أقل من قيمة المعلمة max_connections.

المعلمات الأخرى التي نوقشت في الخطوات السابقة هي كما يلي:

- replication_mode: تحول هذه المعلمة النسخ بشكل صريح إلى وضع التشغيل. افتراضيا، يكون في وضع الإيقاف.
- load_balance_on: تفعيل هذه المعلمة يضمن أن pgpool يقسم الحمل إلى كافة المضيفين أو العقد المرفقة بالنظام.
- master_slave_mode: تفعّل هذه المعلمة وضع التابع/ المتبوع. يجب جعل هذه المعلمة في وضع الإيقاف عندما يكون replication_mode في وضع التشغيل.

المعلمات الأخرى تأخذ القيم الافتراضية، ويمكنك الرجوع إلى روابط pgpool التالية لمزيد من المعلومات المتعلقة بها:

```
http://www.pgpool.net/docs/latest/pgpool-en.html#config
http://www.pgpool.net/pgpool-web/contrib_docs/simple_sr_setting_3.1/pgpool.conf
```

وأخيرا، بمجرد ضبط المعلمات، فقد حان الوقت لتشغيل pgpool وإجراء اتصالات على منفذ 9999 pgpool.

بمجرد تشغيل pgpool، يمكننا أن نرى أن بعض العمليات في الخلفية قد بدأت بالفعل، كما هو موضح في الخطوة ٥ من القسم السابق.

قبل إجراء اتصالات باستخدام pgpool، نستخدم أساسا أداة تسمى pcp_node_info للتحقق من حالة العقد، كما هو موضح في الخطوة ٦ من القسم السابق.

يحتوي الأمر pcp_node_info على الصياغة التالية:

```
pcp_node_info <timeout> <hostname> <port> <username> <password> <nodeid>.
```

هنا مقتطف من الخطوة ٦ من القسم السابق:

```
-bash-4.1$ pcp_node_info 5 localhost 9898 postgres postgres 0
localhost 5432 1 0.500000
-bash-4.1$ pcp_node_info 5 localhost 9898 postgres postgres 1
localhost 5433 1 0.500000
```

وفقا للخطوة ٦ من القسم السابق، نقوم بتحديد قيمة مهلة الاتصال لتكون ٥. يشير اسم المضيف إلى مضيف محلي، متبوعا برقم المنفذ ٥٤٣٢ واسم المستخدم / كلمة المرور الذي يتم تعيينه إلى مستخدم postgres مع كلمة المرور ك postgres. المعلمة النهائية هي nodeid،و التي تم تحديدها بـ ٠ للعقدة الأولى. عقدة ٠ في حالتنا يشير إلى دليل البيانات /var/lib/pgsql/9.3/data. وبالمثل، نستخدم pcp_node_info لتحديد المنفذ ٥٤٣٣ وقيمة nodeid بـ ١ لدليل البيانات /var/lib/pgsql/9.3/data1.

وفي كل مرة يشغل الأمر pcp_node_info، سيستجيب النظام مع الإخراج التالي: اسم المضيف، رقم المنفذ، الحالة، ووزن العقدة.

من بين كل هذه القيم، العمود الثالث، الذي يشير إلى حالة العقدة، هو الأهم. إذا كانت قيمة حالة العمود هي ١، فهذا يعني أن العقدة مشغلة ولكن لا يزال يتعين إجراء اتصالات. إذا كانت قيمة حالة العمود هي ٢، فهذا يعني أن العقدة مشغلة وتجمع الاتصالات. إذا كانت قيمة حالة العمود هي ٣، فهذا يعني أن العقدة معطلة ويجب اتخاذ بعض الإجراءات.

في السيناريو الخاص بنا ، قيمة حالة العمود هو ١ لكلا العقدتين، مما يعني أننا على ما يرام، ويمكننا أن نبدأ في إجراء اتصالات مع pgpool. إذا كانت قيمة حالة العمود هي ٣ ، فستحتاج إلى تفعيل العقدة باستخدام الأداة pcp_attach_node. يحتوي الأمر pcp_attach_node نفس الصياغة كالأمر cp_node_info ويمكن استخدامه كما هو موضح في السطر التالي، على افتراض أن قيمة حالة عمود العقدة هي ٣. دعونا نفترض هذه القيمة لحالة عمود العقدة ١:

```
pcp_attach_node 5 localhost 5433 postgres postgres 1
```

في الخطوة ٧ من القسم السابق، نحن على اتصال بـ pgpool عبر منفذ ٩٩٩٩، إنشاء جدول اسمه emp وإدراج بعض السجلات فيه.

في الخطوة ٨، نقوم باختبار النسخ. يمكننا أن نرى بوضوح أن جدول emp والسجلات المقابلة متوفرة هناك عند إجراء اتصالات إلى المنفذ ٥٤٣٣ ومنفذ ٥٤٣٢. وهذا يؤكد النسخ الناجح باستخدام pgpool.



ارجع إلى روابط الويب التالية لمزيد من التفاصيل حول pgpool:

- http://www.pgpool.net/mediawiki/index.php/Relationship_between_max_pool,_num_init_children,_and_max_connections
- http://www.pgpool.net/docs/latest/pgpool-en.html#connection_pool_mode

بدء و إيقاف pgpool

في هذه الوصفة، سوف نعرض الأوامر التي يمكن استخدامها لبدء وإيقاف pgpool.

الاستعداد للعمل

قبل أن تبدأ pgpool، نحن بحاجة إلى ضبط إعدادات pgpool في pgpool.conf. وقد غطينا ذلك في الوصفة السابقة.

كيف ينجز ذلك...

يمكن بدء تشغيل أداة pgpool بطريقتين:

- من خلال بدء تشغيل خدمة pgpool في سطر الأوامر بصفة المستخدم الجذري:

```
service pgpool start
```

- من خلال تنفيذ الأمر pgpool على الطرفية:

```
pgpool
```

وبالمثل، يمكن وقف pgpool بطريقتين:

- عن طريق إيقاف خدمة pgpool في سطر الأوامر بصفة المستخدم الجذري:

```
service pgpool stop
```

- من خلال تنفيذ الأمر pgpool مع خيار الإيقاف:

```
pgpool stop
```

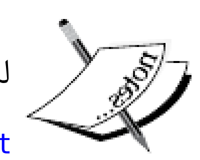
كيف تعمل...

بدء ووقف pgpool بسيط نسبيا، كما رأينا في القسم السابق. ومع ذلك، يأتي pgpool مع الكثير من الخيارات، وفيما يلي الصيغة الأكثر شيوعا لبدء pgpool:

```
pgpool [-c][-f config_file][-a hba_file][-F pcp_config_file]
```

وتناقش هذه الخيارات على النحو التالي:

- c-: يستخدم الخيار c- لمسح ذاكرة التخزين المؤقت للاستعلام.
- fconfig_file-: يحدد هذا الخيار ملف التضبيط pgpool.conf، ويحصل pgpool على ضبطه من هذا الملف عند بدء تشغيله
- a hba_file-: يحدد هذا الخيار ملف الاستيثاق الذي سيستخدم عند بدء تشغيل pgpool
- F pcp_config_file-: يحدد هذا الخيار ملف كلمة المرور، pcp.conf، ليستخدم عند بدء تشغيل pgpool



للحصول على الصياغة الكامل من pgpool، راجع رابط الويب التالي:

http://www.pgpool.net/docs/latest/pgpool-en.html#start

لوقف pgpool، يمكن استخدام نفس الخيارات التي استخدمت في وقت سابق لبدء pgpool. ومع ذلك، جنبا إلى جنب مع هذه المفاتيح، يمكننا أيضا تحديد الوضع الذي يحتاج إلى استخدامها أثناء إيقاف pgpool.

هناك وضعان أين يمكن وقف pgpool:

- **الوضع الذكي**: يتم تحديد هذا الخيار باستخدام الخيار m s (smart)-. في هذا الوضع، ننتظر أولا قطع اتصال العملاء ثم إيقاف تشغيل pgpool.
- **الوضع السريع**: يمكن ضبط هذا الوضع من خلال تحديد الخيار m f (fast)-. في هذا الوضع، pgpool لا تنتظر قطع اتصال العملاء و يقوم بإيقاف تشغيل pgpool على الفور.

الصياغة الكاملة لأمر وقف pgpool هو كما يلي:

```
pgpool [-f config_file][-F pcp_config_file] [-m {s[mart]|f[ast]|]}] stop
```

عادة، إذا كان هناك أي عملاء متصلين،فإن pgpool سينتظر منهم قطع الاتصال ويقوم بالإنهاء من تلقاء نفسه. ومع ذلك، إذا كنت ترغب في إيقاف pgpool قسرا دون انتظار قطع الاتصال من طرف العملاء ، يمكنك استخدام الأمر التالي:

```
pgpool -m fast stop
```

إعداد pgbouncer

في هذه الوصفة، سنعرض الخطوات المطلوبة لتثبيت pgbouncer.

الاستعداد للعمل

يمكننا إما القيام بتثبيت الكامل القائم على المصدر أو استخدام مدير حزم نظام التشغيل الخاصة لتثبيت pgbouncer.

كيف ينجز ذلك...

على نظام أوبنتو أو دبيان، نحن بحاجة لتنفيذ الأمر التالي لتثبيت pgbouncer:

```
apt-get install pgbouncer.
```

على توزيعات لينكس المستندة إلى سينتوس أو فيدورا أو ريد هات يمكننا تنفيذ الأمر التالي:

```
yum install pgbouncer
```

إذا كنت تقوم بتثبيت كامل المصدر، فإن تسلسل الأوامر يكون كما يلي:

١. قم بتنزيل ملف تثبيت الأرشيف من الرابط التالي:

http://pgfoundry.org/projects/pgbouncer

٢. قم بفك الأرشيف الذي نزلته وادخل إلى الدليل المصدر:

```
tar -xzf pgbouncer-1.5.4.tar.gz
cd pgbouncer-1.5.4
```

٣. الخطوة التالية هي بناء ومواصلة تثبيت البرنامج:

```
./configure —prefix=/usr/local
make & make install
```

٤. الآن قم بإنشاء مجلد الإعدادات لوضع ملف ضبط pgbouncer فيه. يمكن استخدام هذا الملف لاحقا لإجراء تغييرات في المعلمة:

```
mkdir /etc/pgbouncer
chown -R postgres:postgres /etc/pgbouncer
```

كيف تعمل...

إذا كنت تستخدم مدير حزم الخاص بنظام التشغيل لتثبيت pgbouncer، فإنه ملفات الضبط وملفات السجل المطلوبة بواسطة pgbouncer ستنشأ تلقائيا. ومع ذلك، إذا كنت ترغب في تثبيت pgbouncer من المصدر، فإن هناك بعض الخطوات الإضافية المطلوبة. سوف تكون بحاجة لإنشاء مجلدات حيث يمكن لـ pgbouncer حفظ سجلات النشاط و ملفات قفل الخدمة فيها. وستكون بحاجة أيضا لإنشاء مجلد الضبط حيث سيحفظ ملف الضبط لـ pgbouncer. كل هذه الخطوات تحتاج إلى أن تنفذ يدويا كما هو مبين في الخطوات ٢ و ٣ و ٤ في القسم السابق.

تجميع الاتصالات باستخدام pgbouncer

في هذه الوصفة، سنذهب لتنفيذ pgbouncer وقياس نتائج أداء اتصالات قاعدة البيانات إلى قاعدة البيانات عبر pgbouncer مقابل اتصالات قاعدة البيانات العادية.

الاستعداد للعمل

قبل أن نقوم بضبط وتنفيذ تجميع الاتصالات، يجب تثبيت الأداة pgbouncer. قمنا بتغطية تثبيت pgbouncer في الوصفة السابقة.

كيف ينجز ذلك...

١. أولا، سنقوم بتعديل بعض إعدادات الضبط في ملف التضبيط pgbouncer.ini، كما يلي. الإدخالات الأولى هما لقواعد البيانات التي ستمرر من خلال pgbouncer. بعد ذلك، نقوم بضبط المعلمة listen_addr إلى *، مما يعني أنها ستنصت إلى جميع عناوين IP. وأخيرا، وضعنا آخر معلمتين، وهما auth_file، موقع ملف الاستيثاق و auth_type، ويشير إلى نوع الاستيثاق المستخدم. نحن نستخدم plain كنوع استيثاق، مما يدل على أننا نستخدم آلية مستندة إلى كلمة المرور هنا للاستيثاق:

```
vi /etc/pgbouncer/pgbouncer.ini
```

```
postgres = host=localhost dbname=postgres
pgtest = host=localhost dbname=pgtest
listen_addr = *
auth_file = /etc/pgbouncer/userlist.txt
auth_type = md5
```

٢. الخطوة التالية هي إنشاء قائمة مستخدمين تحتوي على المستخدمين الذين يسمح لهم بالنفاذ إلى قواعد البيانات من خلال pgbouncer. سيكون تنسيق الإدخالات في قائمة المستخدمين بالشكل التالي اسم مستخدم متبوعا بكلمة مرور المستخدم كما هو موضح في الأمر التالي حيث يكون الإدخال الأول لاسم المستخدم الذي تكون قيمته هي author , والإدخال الثاني لكلمة المرور القيمة هي password.. وبما أننا قمنا بتعيين نوع الاستيثاق كـ MD5، علينا استخدام إدخال كلمة مرور MD5 في قائمة المستخدمين. لو قمنا بتعيين نوع الاستيثاق كـ plain ، فإن كلمة المرور الفعلية يجب توفيرها في قائمة المستخدمين:

```
postgres=# CREATE role author LOGIN PASSWORD 'author' SUPERUSER;
CREATE ROLE
postgres=# select rolname ,rolpassword from pg_authid where rolname='author';
 rolname |             rolpassword
---------+-------------------------------------
 author  | md5d50afb6ec7b2501164b80a0480596ded
(1 row)
```

يمكن تعريف كلمة مرور MD5 التي حصلنا عليها في ملف userlist للمستخدم المناسب:

```
vi /etc/pgbouncer/userlist.txt

"author" "md5d50afb6ec7b2501164b80a0480596ded"
```

٣. بمجرد أن قمنا بضبط ملف التضبيط pgbouncer.ini وإنشاء ملف userlist، فإن الخطوة التالية ستكون بدء خدمة pgbouncer:

```
service pgbouncer start
```

٤. بمجرد تشغيل خدمة pgbouncer ، ستكون الخطوة التالية هي إجراء اتصالات معها. افتراضيا، تشغيل خدمة pgbouncer على المنفذ ٦٤٣٢، لذلك أية اتصالات إلى خدمة pgbouncer يجب أن تكون على المنفذ ٦٤٣٢:

```
psql -h localhost -p 6432 -d postgres -U author -W
```

٥. الآن بعد أن قمنا بإجراء اتصالات باستخدام pgbouncer، الخطوة المنطقية التالية هي معرفة ما إذا كان هناك أي تحسينات في الأداء باستخدام pgbouncer. لهذا الغرض، سنقوم بإنشاء قاعدة بيانات مؤقتة - تلك التي عرفت في البداية في ملف pgbouncer.ini - وإدراج السجلات فيها، ثم قياس الاتصالات التي أجريت مقارنة بقاعدة البيانات هذه:

```
createdb pgtest
pgbench -i -s 10 pgtest
```

٦. نحن نقوم بقياس النتائج مقارنة بقاعدة البيانات pgtest:

```
-bash-3.2$ pgbench -t 1000 -c 20 -C -S pgtest
starting vacuum...end.
transaction type: SELECT only
scaling factor: 10
query mode: simple
number of clients: 20
number of threads: 1
number of transactions per client: 1000
number of transactions actually processed: 20000/20000
tps = 217.374571 (including connections establishing)
tps = 1235.875488 (excluding connections establishing)
```

٧. والخطوة الأخيرة ستكون لقياس النتائج مقارنة بقاعدة البيانات pgtest على منفذ 6432 pgbouncer:

```
-bash-3.2$ pgbench -t 1000 -c 20 -C -S pgtest -p 6432 -U author
Password:
starting vacuum...end.
transaction type: SELECT only
scaling factor: 10
query mode: simple
number of clients: 20
number of threads: 1
number of transactions per client: 1000
number of transactions actually processed: 20000/20000
tps = 2033.768075 (including connections establishing)
tps = 53124.095230 (excluding connections establishing)
```

كيف تعمل...

هنا يمكننا أن نرى بضعة أشياء. في البداية، قمنا بضبط ملف التضبيط pgbouncer.ini جنبا إلى جنب مع ملف userlist، والذي سيستخدمان للوصول إلى قواعد البيانات عبر pgbouncer. ويمكن رؤية فعالية pgbouncer في الخطوتين ٦ و ٧ في القسم السابق. يمكننا أن نرى أن زيادة الإنتاجية إلى ٢٠٣٣.٧٦٨ معاملة في الثانية الواحدة عندما يستخدم pgbouncer، في حين عندما لا يستخدم pgbouncer، تنخفض الإنتاجية إلى ٢١٧.٣٧ معاملة فقط في الثانية الواحدة. في الواقع، استخدام pgbouncer يزيد الإنتاجية بنسبة ١٠ مرات تقريبا.

هناك المزيد...

في المقطع كيف تعمل ... قمنا بضبط معلمات في ملف التضبيط pgbouncer.ini. ومع ذلك، هناك العديد من المعلمات التي يمكن ضبطها، وإذا لم تضبط، فإنها سوف تأخذ الإعدادات الافتراضية. يرجى الرجوع إلى الرابط التالي للحصول على مزيد من التفاصيل حول معلمات pgbouncer:

http://pgbouncer.projects.pgfoundry.org/doc/config.html

إدارة pgbouncer

توفر الأداة المساعدة pgbouncer وحدة تحكم إدارية لعرض حالة تجمع واتصالات العميل. في هذه الوصفة، سنعرض المعلومات المتعلقة باتصالات pgbouncer (اتصالات العميل)، وعرض حالة التجميع، والحصول على إحصاءات تجميع الاتصالات.

الاستعداد للعمل

قبل أن نصدر أي أوامر، نحن بحاجة أولا للاتصال بوحدة التحكم الإدارية لـ pgbouncer. لهذا الغرض، نحن بحاجة إلى تعيين المعلمة admin_users في ملف التضبيط pgbouncer.ini:

```
vi /etc/pgbouncer/pgbouncer.ini

admin_users = author
```

بمجرد حفظ التغييرات السابقة في ملف التضبيط pgbouncer.ini، تحتاج خدمة pgbouncer إلى إعادة تشغيل لضمان أن تغييرات المعلمة نافذة المفعول:

```
service pgbouncer restart
```

بمجرد القيام بذلك، يمكننا إجراء اتصالات بوحدة تحكم إدارة pgbouncer عبر الأمر التالي:

```
psql -p 6432 -U author pgbouncer
```

كيف ينجز ذلك...

مع مساعدة من وحدة تحكم إدارة pgbouncer، يمكننا الحصول على معلومات بخصوص العملاء والخوادم، وصحة التجميع:

١- للحصول على معلومات حول العملاء، أصدر أمر SHOW CLIENTS ، كما هو موضع في اللقطة التالية، في محث إدارة pgbouncer :

```
pgbouncer=# show clients;
 type |  user  | database  |  state  | addr | port | local_addr | local_port |    connect_time     |    request_time     |    ptr    |   link
------+--------+-----------+---------+------+------+------------+------------+---------------------+---------------------+-----------+-----------
 C    | author | pgbouncer | active  | unix | 6432 | unix       |       6432 | 2014-11-21 18:05:43 | 2014-11-21 18:06:44 | 0x9e1ca50 |
 C    | author | pgtest    | active  | unix | 6432 | unix       |       6432 | 2014-11-21 18:06:42 | 2014-11-21 18:06:42 | 0x9e1d8d0 | 0x9e00518
 C    | author | pgtest    | active  | unix | 6432 | unix       |       6432 | 2014-11-21 18:06:42 | 2014-11-21 18:06:42 | 0x9e1d448 | 0x9e00090
 C    | author | pgtest    | active  | unix | 6432 | unix       |       6432 | 2014-11-21 18:06:42 | 2014-11-21 18:06:42 | 0x9e1d9b8 | 0x9e006e8
 C    | author | pgtest    | active  | unix | 6432 | unix       |       6432 | 2014-11-21 18:06:42 | 2014-11-21 18:06:42 | 0x9e1daa0 | 0x9e007d0
 C    | author | pgtest    | active  | unix | 6432 | unix       |       6432 | 2014-11-21 18:06:42 | 2014-11-21 18:06:42 | 0x9e1d7e8 | 0x9dfffa8
 C    | author | pgtest    | active  | unix | 6432 | unix       |       6432 | 2014-11-21 18:06:42 | 2014-11-21 18:06:42 | 0x9e1cd08 | 0x9e00430
 C    | author | pgtest    | active  | unix | 6432 | unix       |       6432 | 2014-11-21 18:06:42 | 2014-11-21 18:06:42 | 0x9e1db88 | 0x9dff868
 C    | author | pgtest    | active  | unix | 6432 | unix       |       6432 | 2014-11-21 18:06:42 | 2014-11-21 18:06:42 | 0x9e1d530 | 0x9e00178
 C    | author | pgtest    | active  | unix | 6432 | unix       |       6432 | 2014-11-21 18:06:42 | 2014-11-21 18:06:42 | 0x9e1d618 | 0x9e00260
 C    | author | pgtest    | active  | unix | 6432 | unix       |       6432 | 2014-11-21 18:06:42 | 2014-11-21 18:06:42 | 0x9e1dc70 | 0x9e00348
 C    | author | pgtest    | active  | unix | 6432 | unix       |       6432 | 2014-11-21 18:06:42 | 2014-11-21 18:06:42 | 0x9e1d700 | 0x9e009a0
 C    | author | pgtest    | active  | unix | 6432 | unix       |       6432 | 2014-11-21 18:06:42 | 2014-11-21 18:06:42 | 0x9e1d190 | 0x9e008b8
```

١. للحصول على معلومات حول اتصالات الخادم، قم بإصدار أمر SHOW SERVERS، كما هو موضح في لقطة الشاشة التالية، في محث إدارة pgbouncer:

```
pgbouncer=# show servers;
 type |  user  | database | state  |   addr    | port | local_addr | local_port |    connect_time     |    request_time     |    ptr    |   link
------+--------+----------+--------+-----------+------+------------+------------+---------------------+---------------------+-----------+-----------
 S    | author | pgtest   | active | 127.0.0.1 | 5432 | 127.0.0.1  |      56002 | 2014-11-21 18:05:57 | 2014-11-21 18:06:42 | 0x9e00518 | 0x9e1d8d0
 S    | author | pgtest   | active | 127.0.0.1 | 5432 | 127.0.0.1  |      55997 | 2014-11-21 18:05:57 | 2014-11-21 18:06:42 | 0x9e00090 | 0x9e1d448
 S    | author | pgtest   | active | 127.0.0.1 | 5432 | 127.0.0.1  |      56004 | 2014-11-21 18:05:57 | 2014-11-21 18:06:42 | 0x9e006e8 | 0x9e1d9b8
 S    | author | pgtest   | active | 127.0.0.1 | 5432 | 127.0.0.1  |      56005 | 2014-11-21 18:05:57 | 2014-11-21 18:06:42 | 0x9e007d0 | 0x9e1daa0
 S    | author | pgtest   | active | 127.0.0.1 | 5432 | 127.0.0.1  |      55996 | 2014-11-21 18:05:57 | 2014-11-21 18:06:42 | 0x9dfffa8 | 0x9e1d7e8
 S    | author | pgtest   | active | 127.0.0.1 | 5432 | 127.0.0.1  |      56001 | 2014-11-21 18:05:57 | 2014-11-21 18:06:42 | 0x9e00430 | 0x9e1cd08
 S    | author | pgtest   | active | 127.0.0.1 | 5432 | 127.0.0.1  |      55988 | 2014-11-21 18:05:57 | 2014-11-21 18:06:42 | 0x9dff868 | 0x9e1db88
 S    | author | pgtest   | active | 127.0.0.1 | 5432 | 127.0.0.1  |      55998 | 2014-11-21 18:05:57 | 2014-11-21 18:06:42 | 0x9e00178 | 0x9e1d530
 S    | author | pgtest   | active | 127.0.0.1 | 5432 | 127.0.0.1  |      55999 | 2014-11-21 18:05:57 | 2014-11-21 18:06:42 | 0x9e00260 | 0x9e1d618
 S    | author | pgtest   | active | 127.0.0.1 | 5432 | 127.0.0.1  |      56000 | 2014-11-21 18:05:57 | 2014-11-21 18:06:42 | 0x9e00348 | 0x9e1dc70
 S    | author | pgtest   | active | 127.0.0.1 | 5432 | 127.0.0.1  |      56007 | 2014-11-21 18:05:57 | 2014-11-21 18:06:42 | 0x9e009a0 | 0x9e1d700
```

٢. وبالمثل، يمكنك إصدار أوامر SHOW POOLS و SHOW STATS للحصول على معلومات بشأن حالة التجميع وإحصاءات التجميع على التوالي، كما هو مبين في الصورة التالية:

```
pgbouncer=# show pools;
 database  |   user    | cl_active | cl_waiting | sv_active | sv_idle | sv_used | sv_tested | sv_login | maxwait
-----------+-----------+-----------+------------+-----------+---------+---------+-----------+----------+---------
 pgbouncer | pgbouncer |         1 |          0 |         0 |       0 |       0 |         0 |        0 |       0
 pgtest    | author    |        20 |          1 |        20 |       0 |       0 |         0 |        0 |      85
(2 rows)

pgbouncer=# show stats;
 database  | total_requests | total_received | total_sent | total_query_time | avg_req | avg_recv | avg_sent | avg_query
-----------+----------------+----------------+------------+------------------+---------+----------+----------+-----------
 pgbouncer |              1 |              0 |          0 |                0 |       0 |        0 |        0 |         0
 pgtest    |           8131 |         442818 |     536895 |         16433811 |       0 |        0 |        0 |         0
(2 rows)
```

كيف تعمل...

كما رأينا في القسم السابق، يمكننا الحصول على معلومات حول العملاء والخوادم، وصحة التجميع والإحصاءات. عند إصدار الأمر SHOW CLIENTS في محث إدارة pgbouncer فإن pgbouncer يوفر لك قائمة من العملاء التي كانت إما تستخدم اتصال بوستجريسكل أو في انتظار ذلك. سنناقش بعض الأعمدة الهامة المعروضة في إخراج أمر SHOW CLIENTS هنا:

- user: تعرض القيمة في هذا العمود المستخدم المتصل بقاعدة البيانات.
- database: تعرض القيمة في هذا العمود اسم قاعدة البيانات التي يوصل العميل بها.
- state: تعرض القيمة هنا حالة جلسة المستخدم المتصل حاليا. يمكن أن يكون اتصال العميل في حالة نشطة أو مستعملة أو في انتظار أو حالة خمول.
- connect_time: تشير القيمة في هذا العمود إلى الوقت الذي بدأ به pgbouncer اتصال العميل ببوستجريسكل.
- request_time: توضح قيمة هذا العمود الطابع الزمني لأحدث طلب للعميل.
- port: تشير القيمة في هذا العمود إلى المنفذ الذي يتصل به العميل.

لدينا أمر SHOW SERVERS، والذي يستخدم لعرض معلومات حول كل اتصال يتم استخدامه لتلبية طلبات العميل. تحتوي مخرجات SHOW SERVERS على أعمدة مماثلة، شرحناها في SHOW CLIENTS. والفرق الوحيد هو عمود النوع type. إذا كانت قيمة عمود النوع هي S، فهذا يعني أنه إدخال الخادم. إذا كانت قيمة عمود النوع هي C، فهذا يعني أنه إدخال عميل. وتناقش بعض الأعمدة الهامة الأخرى للإخراج SHOW SERVERS كما يلي:

- user: تعرض القيمة في هذا العمود المستخدم المتصل بقاعدة البيانات.
- database: تعرض القيمة في هذا العمود اسم قاعدة البيانات التي يوصل الاتصال بها.

- state: القيمة هنا تعرض حالة اتصال خادم pgbouncer. يمكن أن تكون حالة الخادم نشطة أو مستخدمة أو خاملة.
- connect_time: تشير القيمة في هذا العمود إلى الوقت الذي أجري فيه الاتصال.
- request_time: تعرض قيمة هذا العمود الطابع الزمني عند إصدار آخر طلب.
- port: تشير القيمة في هذا العمود إلى رقم المنفذ لخادم بوستجريسكل.

يعرض الأمر SHOW POOLS صف لكل قاعدة بيانات يعمل فيها pgbouncer كوسيط. بعض الأعمدة الهامة في إخراج SHOW POOLS هي كما يلي:

- cl_active: تعرض القيمة في هذا العمود عدد العملاء الناشطين حاليا وقموا بتعيين اتصال بالخادم.
- cl_waiting: تعرض القيمة في هذا العمود عدد العملاء الذين ينتظرون اتصال الخادم.
- sl_active: تعرض القيمة في هذا العمود عدد اتصالات الخادم التي عينت إلى عملاء pgbouncer.
- sl_idle: تعرض القيمة هنا عدد اتصالات الخادم الخاملة، بما في ذلك تلك التي لا تكون قيد الاستخدام.
- sl_used: تعرض القيمة في هذا العمود عدد اتصالات الخادم المستخدمة. في الواقع، هذه الاتصالات هي في الواقع خاملة ولكن لم يضع pgbouncer علامة عليها لإعادة استخدامها حتى الآن.

يعرض الأمر SHOW STATS إحصائيات تجمع الاتصالات ذات الصلة المتعلقة ب pgbouncer لقواعد البيانات التي يعمل pgbouncer كوسيط فيها. بعض الأعمدة الهامة في ناتج عرض الإحصائيات هي كما يلي:

- total_requests: تعرض القيمة في هذا العمود العدد الإجمالي لطلبات SQL المجمعة بواسطة pgbouncer
- total_received: تعرض القيمة في هذا العمود إجمالي حجم حركة مرور الشبكة (المقاسة بالبايتات) التي استلمت بواسطة pgbouncer
- total_sent: تعرض قيمة هذا العمود إجمالي حجم حركة مرور الشبكة (المقاسة بالبايتات) التي أرسلت بواسطة pgbouncer
- total_query_time: القيمة في هذا العمود تعرض مقدار الوقت بالميكروثانية التي أنفقها pgbouncer للتواصل مع العميل في هذا التجمع.

٩- تقسيم الجدول

في هذا الفصل، سنغطي الوصفات التالية:

- تنفيذ التقسيم
- إدارة الأقسام
- التقسيم وقيود الاستبعاد
- طرق التقسيم البديلة
- تثبيت PL/Proxy
- التقسيم مع PL/Proxy

مقدمة

يعرف تقسيم الجداول على أنه تجزئة جدول كبير إلى أجزاء أصغر. ويدعم بوستجريسكل تقسيم الجدول الرئيسي. ويوفر تقسيما واضحا بشكل تام للتطبيقات إذا نفذ بشكل صحيح. وللتقسيم الكثير من الفوائد، التي سنذكرها فيما يلي:

- تحسين أداء الاستعلام بشكل ملحوظ لأنواع معينة من الاستعلامات.
- يمكن أن يؤدي التقسيم إلى تحسين أداء التحديث. كلما قامت الاستعلامات بتحديث جزء كبير من قسم واحد، يمكن تحسين الأداء عن طريق إجراء تفحص متسلسل لهذا القسم، بدلا من استخدام القراءات والكتابات العشوائية المبعثرة عبر الجدول بأكمله.
- يمكن إنجاز عملات التحميل المجمعة وعمليات الحذف عن طريق إضافة أو إزالة الأقسام إذا ضمنت المتطلبات في تصميم التقسم. كما تعمل عمليات ALTER TABLE NO INHERIT و DROP TABLE بشكل أسرع في حالة التقسيم من العملية المجمعة. بالإضافة إلى أن هذه الأوامر تجنب عمليات التنظيف من الوقوع في التراكم العالي الناجم عن الحذف بالجملة.
- يمكن إرسال البيانات قليلة الاستخدام إلى وسيط أرخص وأبطأ.

تنفيذ التقسيم

سنقوم هنا بتغطية عملية تقسيم الجدول وشرح الخطوات التي تحتاج إليها من أجل التقسيم.

الاستعداد للعمل

المعرفة المسبقة بتصميم قاعدة البيانات والتطبيع هو الشرط الوحيد.

كيف ينجز ذلك...

يجب تنفيذ سلسلة من الخطوات من أجل إعداد تقسيم الجدول:

١. أولا، قم بإنشاء جدول رئيسي مع كافة الحقول. الجدول الرئيسي هو الجدول الذي سيستخدم كقاعدة لتقسيم البيانات إلى جداول أخرى، أي الأقسام. وإنشاء الفهرس يكون اختياري بالنسبة للجدول الرئيسي؛ ومع ذلك، فهناك فوائد تدعم الأداء إذا استخدم الفهرس، سنقوم هنا بإنشاء فهرس من منظور تحسين الأداء كما يلي:

```
CREATE TABLE country_log (
    created_at TIMESTAMP WITH TIME ZONE DEFAULT NOW(),
    country_code char(2),
    content text
);
CREATE INDEX country_code_idx ON country_log USING btree (country_code);
```

٢. الخطوة التالية هي إنشاء الجداول الفرعية التي تنحدر من الجدول الرئيسي:

```
CREATE TABLE country_log_ru ( CHECK ( country_code = 'ru') ) INHERITS (country_log);
CREATE TABLE country_log_sa ( CHECK ( country_code = 'sa' ) ) INHERITS (country_log);
```

٣. بعد ذلك، قم بإنشاء فهرس لكل جدول صغير:

```
CREATE INDEX country_code_ru_idx ON country_log_ru USING btree (country_code);
CREATE INDEX country_code_sa_idx ON country_log_sa USING btree (country_code);
```

٤. ثم، قم بإنشاء وظيفة الزناد trigger function مع مساعدة من البيانات التي ستوجه إلى جدول التقسيم المناسب، على النحو التالي:

```
CREATE OR REPLACE FUNCTION country_insert_trig() RETURNS TRIGGER AS $$
BEGIN
    IF ( NEW.country_code = 'ru' ) THEN
        INSERT INTO country_log_ru VALUES (NEW.*);
    ELSIF ( NEW.country_code = 'sa' ) THEN
        INSERT INTO country_log_sa VALUES (NEW.*);
    ELSE
        RAISE EXCEPTION 'Country unknown';
    END IF;
    RETURN NULL;
END;
$$ LANGUAGE plpgsql;
```

٥. الآن، قم بإنشاء الزناد وإرفاق وظيفة الزناد إلى الجدول الرئيسي:

```
CREATE TRIGGER country_insert
BEFORE INSERT ON country_log
FOR EACH ROW EXECUTE PROCEDURE country_insert_trig();
```

٦. بعد ذلك، أدخل البيانات في الجدول الرئيسي، كما هو موضح هنا:

```
postgres=# INSERT INTO country_log (country_code, content) VALUES ('ru',
'content-ru');
postgres=# INSERT INTO country_log (country_code, content) VALUES ('sa',
'content-sa');
```

٧. الخطوة النهائية هي اختيار البيانات من كل من الجداول الرئيسية والفرعية لتأكيد تقسيم البيانات في الجداول الفرعية، على النحو التالي:

```
postgres=# SELECT * from country_log;
          created_at           | country_code |  content
-------------------------------+--------------+------------
 2014-11-30 12:10:06.123189-08 | ru           | content-ru
 2014-11-30 12:10:14.22666-08  | sa           | content-sa
(2 rows)
postgres=# select * from country_log_ru;
          created_at           | country_code |  content
-------------------------------+--------------+------------
 2014-11-30 12:10:06.123189-08 | ru           | content-ru
(1 row)
postgres=# select * from country_log_sa;
          created_at          | country_code |  content
------------------------------+--------------+------------
 2014-11-30 12:10:14.22666-08 | sa           | content-sa
```

```
(1 row)
```

كيف يعمل ...

وفيما يلي شرح مفصل للخطوات التي نفذت في القسم السابق:

- يدعم بوستجريسكل أساسا التقسيم عبر ميراث الجدول. ومن ثم، يقسم الجداول بطريقة تضمن أن يرث كل جدول فرعي من الجدول الأصلي. ولتحقيق هذا الغرض، نقوم بإنشاء جدولين فرعيين، أي country_log_ru و country_log_sa، في الخطوة ٢ من القسم السابق. وترث هذه الجداول الفرعية من الأصل أو الجدول الرئيسي، country_log باستخدام الكلمة المفتاحية INHERITS في الجدول الرئيسي لعبارة CREATE TABLE DDL لكل من الجداول الفرعية. كانت هذه الخطوة الأولي.

- الخطوة التالية، من السيناريو هي بناء التقسيم بحيث تخزن السجلات حسب البلد في جدول خاص بالبلد. كانت الحالة التي استخدمناها في القسم السابق هي التأكد من أن جميع سجلات روسيا تذهب في الجدول country_log_ru وجميع سجلات جنوب أفريقيا تذهب في الجدول country_log_sa. ولتحقيق هذا الهدف، نقوم بتعريف وظيفة الزناد country_insert_trig، مما يساعد على تقسيم البيانات إلى جدول خاص بكل بلد كلما تم تشغيل عبارة INSERT على الجدول الرئيسي للبلد. في اللحظة التي تنفذ فيها عبارة INSERT على الجدول country_log، فإن الزناد country_log يقدح في ما يسمى () country_insert_trig. تقوم وظيفة الزناد () country_insert_trig بفحص السجلات المدرجة، وإذا وجدت سجلات لروسيا (بفحص العبارة الشرطية 'NEW.country_code = 'ru) في جدول country_log، ثم تدرج السجل المذكور في الجدول الفرعي country_log_ru. إذا كان السجل المدرج في الجدول الرئيسي لبلد مخصص كجنوب إفريقيا (NEW country_code = 'sa') سيقوم السجل بتسجيل نفسه في جدول فرعي country_log_sa. ووظيفة الزناد هي تقسيم البيانات بهذه الطريقة. يستخدم القسم التالي من التعليمات البرمجية في الدالة الزناد () country_insert_trig المنطق المحدد في شرط IF لتقسيم البيانات إلى الجداول الفرعية:

```
IF ( NEW.country_code = 'ru' ) THEN
  INSERT INTO country_log_ru VALUES (NEW.*);
ELSIF ( NEW.country_code = 'sa' ) THEN
  INSERT INTO country_log_sa VALUES (NEW.*);
ELSE
  RAISE EXCEPTION 'Country unknown';
END IF;
```

- وأخيرا، بمجرد تقسيم البيانات إلى جداول فرعية، فإن الخطوة الأخيرة هي التحقق من ذلك من خلال مقارنة السجلات بين الجداول الفرعية والجدول الرئيسي، كما هو موضح في الخطوة ٧.

في البداية، أدرج سجلان في الجدول الرئيسي country_log. ويمكن تأكيد ذلك عن طريق تشغيل الاستعلام SELECT على جدول country_log. هنا، يمكننا أن نرى سجلين في جدول country_log، واحد لروسيا، التي حددها رمز البلد ru، والآخر لجنوب أفريقيا، التي حددها رمز البلد sa :

```
postgres=# SELECT * from country_log;
          created_at           | country_code |  content
-------------------------------+--------------+------------
 2014-11-30 12:10:06.123189-08 | ru           | content-ru
 2014-11-30 12:10:14.22666-08  | sa           | content-sa
(2 rows)
```

الخطوة التالية هي تشغيل استعلامات SELECT على الجداول المتفرعة من الجدول الرئيسي، وهي The next country_log_ru و country_log_sa:

```
postgres=# select * from country_log_ru;
          created_at           | country_code |  content
-------------------------------+--------------+------------
 2014-11-30 12:10:06.123189-08 | ru           | content-ru
(1 row)
postgres=# select * from country_log_sa;
          created_at          | country_code |  content
------------------------------+--------------+------------
 2014-11-30 12:10:14.22666-08 | sa           | content-sa
(1 row)
```

من الناتج السابق، يمكنك أن ترى أن هناك سجل واحد فقط في كل جدول فرعي country_log_ru و country_log_sa. والواقع أن وظيفة الزناد ()country_insert_trig قد قسمت بيانات السجل في الجداول الخاصة بكل بلد. تدخل إدخالات عمود country_code مع قيمة Ru، أي بالنسبة لروسيا، في جدول country_log_ru، وإدخالات عمود country_code بقيمة Sa، أي جنوب أفريقيا، انتقل إلى جدول الفرعي country_log_sa.

هناك المزيد

يمكنك الرجوع إلى الروابط التالية للحصول على شرح أكثر تفصيلا حول كيفية تنفيذ التقسيم:

```
https://blog.engineyard.com/2013/scaling-postgresql-performance-table-partitioning
http://www.postgresql.org/docs/9.3/static/ddl-partitioning.html
```

إدارة الأقسام

هنا، سنقوم بعرض لك كيف تبقى مخطط التقسيم سليمة عندما حذف قسم موجود أو إضافة قسم جديد.

الاستعداد للعمل

يرجى الرجوع إلى الوصفة الأولى، تنفيذ التقسيم، قبل أن تقرأ الخطوات المبينة في هذه الوصفة.

كيف ينجز ذلك...

هناك سيناريوهان هنا. يُظهر السيناريو الأول ما يحدث عند حذف قسم موجود، والآخر يظهر ما يحدث عند إضافة قسم جديد. دعونا نناقش كلتا الحالتين.

في السيناريو الأول، سنقوم بحذف جدول قسم موجود. وهنا، سيحذف الجدول country_code_sa، على النحو التالي:

١. قبل حذف الجدول الفرعي country_log_sa، راجع السجلات في الجدول الرئيسي للبلد country_log:

```
postgres=# SELECT * from country_log;
          created_at           | country_code |  content
-------------------------------+--------------+------------
 2014-11-30 12:10:06.123189-08 | ru           | content-ru
 2014-11-30 12:10:14.22666-08  | sa           | content-sa
(2 rows)
```

٢. بعد ذلك، قم بإسقاط الجدول الفرعي country_log_sa، كما هو موضح هنا:

```
postgres=# drop table country_log_sa;
DROP TABLE
```

٣- مرة أخرى، كخطوة أخيرة، إعادة فحص البيانات في الجدول الرئيسي، country_log، بمجرد إسقاط country_log_sa:

```
postgres=# select * from country_log;
           created_at          | country_code |  content
------------------------------+--------------+------------
2014-11-30 14:41:40.742878-08 | ru           | content-ru
```

في السيناريو الثاني، سوف نقوم بإضافة قسم جديد، country_log_default. وفكرة إنشاء هذا القسم هي أنه إذا كانت هناك جداول لا تحدد فيها وظيفة الزناد أي رموز للبلدان، ينبغي أن تدخل تلك السجلات في قسم جدول افتراضي على النحو التالي:

١. قبل إنشاء جدول فرعي، فلنشاهد السجلات الحالية في الجدول الرئيسي لبلدك:

```
postgres=# select * from country_log;
         created_at            | country_code |  content
----------------------------+------------+------------
 2014-11-30 14:41:40.742878-08 | ru            | content-ru
```

٢. بعد ذلك، أنشئ جدولا لعبارة country_log_sa وأنشئ فهرسا على الجدول الفرعي، كما هو موضح هنا:

```
postgres=# CREATE TABLE country_log_default ( ) INHERITS (country_log);
CREATE TABLE
postgres=# CREATE INDEX country_code_default_idx ON country_log_default USING
btree (country_code);
CREATE INDEX;
```

٣. قم بتعديل وظيفة الزناد الموجودة لديك من أجل تحديد شرط لإدراج سجلات السجل لتلك البلدان التي لم يعرف رمز البلد لها بشكل صريح، من أجل إدراجها في جدول سجل محدد لرمز البلد:

```
CREATE OR REPLACE FUNCTION country_insert_trig() RETURNS TRIGGER AS $$
BEGIN
    IF ( NEW.country_code = 'ru' ) THEN
        INSERT INTO country_log_ru VALUES (NEW.*);
    ELSIF ( NEW.country_code = 'sa' ) THEN
        INSERT INTO country_log_sa VALUES (NEW.*);

    ELSE
        INSERT INTO country_log_default VALUES (NEW.*);
    END IF;
    RETURN NULL;
END;
$$ LANGUAGE plpgsql;
```

٤. الآن، قم بإدراج السجلات في الجدول الرئيسي:

```
postgres=# INSERT INTO country_log (country_code, content) VALUES ('dk',
'content-dk');
INSERT 0 0
postgres=# INSERT INTO country_log (country_code, content) VALUES ('us',
'content-us');
INSERT 0 0
```

٥- دعونا نتحقق من السجلات التي أنشئت حديثا في الجدول الرئيسي لبلدك، ونرى ما إذا كانت هذه السجلات قد قسمت إلى جدول country_log_default:

```
postgres=# select * from country_log;
          created_at            | country_code |  content
--------------------------------+--------------+------------
 2014-11-30 14:41:40.742878-08 | ru           | content-ru
 2014-11-30 15:10:28.921124-08 | dk           | content-dk
 2014-11-30 15:10:42.97714-08  | us           | content-us

postgres=# select * from country_log_default;
          created_at            | country_code |  content
--------------------------------+--------------+------------
 2014-11-30 15:10:28.921124-08 | dk           | content-dk
 2014-11-30 15:10:42.97714-08  | us           | content-us
```

كيف يعمل ...

أولا، دعونا نناقش السيناريو الأول حيث نسقط جدول تقسيم الفرعية، country_log_sa. في ما يلي مقتطف الشفرة عرض مسبقا في القسم السابق:

```
postgres=# SELECT * from country_log;
          created_at            | country_code |  content
--------------------------------+--------------+------------
 2014-11-30 12:10:06.123189-08 | ru           | content-ru
 2014-11-30 12:10:14.22666-08  | sa           | content-sa
(2 rows)
postgres=# drop table country_log_sa;
DROP TABLE
postgres=# select * from country_log;
          created_at            | country_code |  content
--------------------------------+--------------+------------
 2014-11-30 14:41:40.742878-08 | ru           | content-ru
```

إذا كنت رجعت إلى تسلسل الأحداث في المخرجات السابقة، يمكنك أن ترى بوضوح أنه بمجرد أن حذف الجدول الفرعي، country_log_sa، تمت إزالة الإدخال المقابل له من الجدول الرئيسي لسجل البلد. هذه التقنية تساعد حقا إذا كان هناك عدد كبير من السجلات بحاجة إلى الحذف من الجدول الرئيسي مرة واحدة حينما يحذف الجدول الفرعي

المقابل. هذا الإجراء هو تلقائي ولا يتطلب تدخل مدير قاعدة البيانات. وبهذه الطريقة، يمكن إدارة هيكل التقسيم والبيانات بسهولة والتعامل معها إذا حذف أي قسم موجود.

وبالمثل، يمكن إدارة البيانات بسهولة عند إضافة قسم جديد. إذا كنت رجعت إلى الخطوة ٢ من السيناريو الثاني في قسم كيفية إجراء ذلك ... يمكنك أن ترى أننا ننشئ جدولا فرعيا جديدا، country_log_default، ويرث من الجدول الرئيسي لبلدك. وبمجرد تعديل دالة الزناد الحالية، country_insert_trig ()، لتشمل الإدخال الذي يحتوي على شرط لتقسيم البيانات إلى القسم الذي أنشئ حديثا، country_log_default ، تشغل عبارة INSERT على الجدول الرئيسي country_log، وإذا كان الشرط مستوفيا لإدراج السجل في الجدول الفرعي country_log_default، فإن ذلك السجل سيدرج في الجدول الفرعي country_log_default. يمكن رؤية ذلك في الخطوات من ٢ إلى ٥ من السيناريو الثاني في قسم كيف ينجز ذلك... عند إضافة قسم.

هناك المزيد

للحصول على شرح أكثر تفصيلا عن التقسيم، يفضل زيارة

http://www.postgresql.org/docs/9.3/static/ddl-partitioning.html

التقسيم و استبعاد القيود

في هذه الوصفة، سوف نتحدث عن استبعاد القيود constraint exclusion وكيف زساعد على تحسين الأداء.

الاستعداد للعمل

إن الألفة مع تقسيم الجدول مطلوب لهذه الوصفة.

كيف ينجز ذلك...

يمكن تفعيل استبعاد القيود بالأمر التالي:

```
SET constraint_exclusion = ON ;
```

كيف يعمل ...

الآن، دعنا نناقش استبعاد القيود.

استبعاد القيود هو في الأساس تقنية تحسين الاستعلام التي تساعد على تحسين أداء الجداول المقسمة.

دعونا نقوم بتحليل خطة الاستعلام لطلب البحث التالي:

```
postgres=# EXPLAIN ANALYZE SELECT * FROM country_log WHERE country_code = 'ru';
                                QUERY PLAN
-----------------------------------------------------------------------------
 Result  (cost=0.00..38.29 rows=16 width=52) (actual time=26.442..27.298 rows=1 loops=1)
   ->  Append  (cost=0.00..38.29 rows=16 width=52) (actual time=26.437..27.289 rows=1
loops=1)
          ->  Seq Scan on country_log  (cost=0.00..0.00 rows=1 width=52) (actual
time=0.002..0.002 rows=0 loops=1)
                 Filter: (country_code = 'ru'::bpchar)
          ->  Bitmap Heap Scan on country_log_ru country_log  (cost=4.29..12.76 rows=5
width=52) (actual time=26.431..26.433 rows=1 loops=1)
                 Recheck Cond: (country_code = 'ru'::bpchar)
                 ->  Bitmap Index Scan on country_code_ru_idx  (cost=0.00..4.29 rows=5
width=0) (actual time=26.413..26.413 rows=1 loops=1)
                        Index Cond: (country_code = 'ru'::bpchar)
          ->  Bitmap Heap Scan on country_log_au country_log  (cost=4.29..12.76 rows=5
width=52) (actual time=0.822..0.822 rows=0 loops=1)
                 Recheck Cond: (country_code = 'ru'::bpchar)
                 ->  Bitmap Index Scan on country_code_au_idx  (cost=0.00..4.29 rows=5
width=0) (actual time=0.817..0.817 rows=0 loops=1)
                        Index Cond: (country_code = 'ru'::bpchar)
          ->  Bitmap Heap Scan on country_log_default country_log  (cost=4.29..12.76
rows=5 width=52) (actual time=0.023..0.023 rows=0 loops=1)
                 Recheck Cond: (country_code = 'ru'::bpchar)
                 ->  Bitmap Index Scan on country_code_default_idx  (cost=0.00..4.29
rows=5 width=0) (actual time=0.013..0.013 rows=0 loops=1)
                        Index Cond: (country_code = 'ru'::bpchar)
 Total runtime: 27.442 ms
(17 rows)
```

إذا قمت بتحليل خطة الاستعلام السابقة لاستعلام SELECT ، ستجد أن الاستعلام يدقق كل قسم من أقسام country_log. هذا السلوك ليس هو الأمثل من منظور أداء الاستعلام.

للتعامل مع هذا السيناريو، يمكننا تفعيل استبعاد القيود. من خلال القيام بذلك، فإن مخطط الاستعلام سيفحص محتويات كل قسم؛ ومع ذلك، فإن المخطط سيحاول إثبات أن التدقيق غير مطلوب للأقسام التي لا تستوفي معايير الاستعلام المحددة في شرط WHERE. عندما يمكن للمخطط إثبات ذلك، فإنه يستبعد هذه الأقسام من خطة الاستعلام. يمكن رؤية ذلك في خطة الاستعلام التي أنشئت للاستعلام بعد تفعيل استبعاد القيود، كما هو موضح هنا:

```
postgres=# SET constraint_exclusion = ON;
```

```
SET
postgres=# EXPLAIN ANALYZE SELECT * FROM country_log WHERE country_code = 'ru';
                           QUERY PLAN
---------------------------------------------------------------------
 Result  (cost=0.00..25.52 rows=11 width=52) (actual time=0.036..0.147 rows=1 loops=1)
   ->  Append  (cost=0.00..25.52 rows=11 width=52) (actual time=0.031..0.138 rows=1
loops=1)
         ->  Seq Scan on country_log  (cost=0.00..0.00 rows=1 width=52) (actual
time=0.003..0.003 rows=0 loops=1)
               Filter: (country_code = 'ru'::bpchar)
         ->  Bitmap Heap Scan on country_log_ru country_log  (cost=4.29..12.76 rows=5
width=52) (actual time=0.025..0.027 rows=1 loops=1)
               Recheck Cond: (country_code = 'ru'::bpchar)
               ->  Bitmap Index Scan on country_code_ru_idx  (cost=0.00..4.29 rows=5
width=0) (actual time=0.017..0.017 rows=1 loops=1)
                     Index Cond: (country_code = 'ru'::bpchar)
         ->  Bitmap Heap Scan on country_log_default country_log  (cost=4.29..12.76
rows=5 width=52) (actual time=0.102..0.102 rows=0 loops=1)
               Recheck Cond: (country_code = 'ru'::bpchar)
               ->  Bitmap Index Scan on country_code_default_idx  (cost=0.00..4.29
rows=5 width=0) (actual time=0.096..0.096 rows=0 loops=1)
                     Index Cond: (country_code = 'ru'::bpchar)
 Total runtime: 0.230 ms
(13 rows)
```

يمكننا أن نرى تحسين الأداء في خطة الاستعلام، حيث يظهر هذا وقت تشغيل إجمالي قدره ٠.٢٣٠ ميلي ثانية، بينما تظهر خطة الاستعلام السابقة إجمالي وقت التشغيل ٢٧.٤٤٢ ميلي ثانية. وهكذا، يمكنك أن ترى فوائد الأداء من خلال تفعيل استبعاد القيود.

أساليب التقسيم البديلة

في هذه الوصفة، سنتحدث عن آلية أخرى يمكن استخدامها لإعادة توجيه INSERTS إلى الأقسام المناسبة. هنا، سوف نتحدث عن استخدام النهج القائم على القواعد بدلا من النهج القائم على الزناد، من أجل إعادة توجيه INSERTS إلى الأقسام المناسبة.

الاستعداد للعمل

الإلمام بتقسيم الجدول مطلوب لهذه الوصفة.

كيف ينجز ذلك...

ما سنفعله الآن هو استخدام نهج قائم على القواعد. للقيام بذلك، قم بتنفيذ الخطوات التالية:

١. لتجنب أي تعارض مع النهج القائم على الزناد المستخدمة سابقا، يجب المضي قدما عن طريق حذف وظيفة الزناد الموجودة، وذلك باستخدام الأمر التالي:

```
postgres=# drop function country_insert_trig() cascade;
```

٢. ستكون الخطوة التالية هي إنشاء قواعد لكل جدول من الجداول الفرعية، بحيث في كل مرة يدرج سجل جديد في الجدول الرئيسي (country_log)،تستدعى القواعد لإعادة توجيه أوامر INSERT إلى جدول القسم المناسب، على النحو الموضح هنا:

```
CREATE RULE country_code_check_ru AS
ON INSERT TO country_log WHERE
    ( NEW.country_code = 'ru' )
DO INSTEAD
    INSERT INTO country_log_ru VALUES (NEW.*);
CREATE RULE country_code_check_sa AS
ON INSERT TO country_log WHERE
    ( NEW.country_code = 'sa' )
DO INSTEAD
    INSERT INTO country_log_sa VALUES (NEW.*);
CREATE RULE country_code_check_default AS
ON INSERT TO country_log WHERE
    ( NEW.country_code != 'ru' OR NEW.country_code != 'sa' )
DO INSTEAD
    INSERT INTO country_log_default VALUES (NEW.*);
```

٣. بعد ذلك، أدخل السجل في الجدول الرئيسي،

```
INSERT INTO country_log (country_code, content) VALUES ('ca', 'content-ca');
```

٤. أخيرا، استخدم الاستعلام SELECT على جدول القسم المعني للتحقق مما إذا أعيد توجيه الأوامر INSERT المستخدمة في الخطوة السابقة إلى جدول القسم المناسب باستخدام النهج القائم على قاعدة كما يلي:

```
postgres=# select * from country_log_default;
          created_at            | country_code |  content
--------------------------------+--------------+------------
 2014-11-30 15:10:28.921124-08 | dk           | content-dk
 2014-11-30 15:10:42.97714-08  | us           | content-us
 2014-12-01 14:36:27.746601-08 | ca           | content-ca
(3 rows)
```

كيف تعمل..

ما نقوم به أساسا هنا هو إنشاء قواعد لجميع الأقسام. والحالة المحددة في القواعد هي نفس الحالة المحددة في دالة الزناد، country_ enter_trig (). دعونا نعرض رمز وظيفة الزناد الذي اشتقت منه الشروط المحددة للقواعد:

```
CREATE OR REPLACE FUNCTION country_insert_trig() RETURNS TRIGGER AS $$
BEGIN
    IF ( NEW.country_code = 'ru' ) THEN
        INSERT INTO country_log_ru VALUES (NEW.*);
    ELSIF ( NEW.country_code = 'sa' ) THEN
        INSERT INTO country_log_sa VALUES (NEW.*);

    ELSE
        INSERT INTO country_log_default VALUES (NEW.*);
    END IF;
    RETURN NULL;
END;
$$ LANGUAGE plpgsql;
```

إذا قمت بإلقاء نظرة على وظيفة الزناد السابقة، فمن الواضح أنه عندما تحتوي البيانات المدرجة في الجدول الرئيسي لبلدك على قيم عمود country_code باسم ru أو sa، فإن الصف المدرج المقابل سيذهب إما إلى جدول country_log_ru أو country_ log_sa ، اعتمادا على إدخال الإدخال country_code المدرج. إذا كانت قيمة العمود country_code التي أدخلت هي أي شيء آخر بخلاف هاتين القيمتين، فستوجه إلى الجدول country_log_default.

وبناء على هذه الشروط، نحدد قواعد جميع الجداول الفرعية، على النحو التالي:

```
CREATE RULE country_code_check_ru AS
ON INSERT TO country_log WHERE
    ( NEW.country_code = 'ru' )
DO INSTEAD
    INSERT INTO country_log_ru VALUES (NEW.*);
CREATE RULE country_code_check_sa AS
ON INSERT TO country_log WHERE
    ( NEW.country_code = 'sa' )
DO INSTEAD
    INSERT INTO country_log_sa VALUES (NEW.*);
CREATE RULE country_code_check_default AS
ON INSERT TO country_log WHERE
    ( NEW.country_code != 'ru' OR NEW.country_code != 'sa' )
DO INSTEAD
    INSERT INTO country_log_default VALUES (NEW.*);
```

حالما تعرف القواعد للجداول الفرعية، فإن الخطوة التالية هي إزالة وظيفة الزناد المستخدمة سابقا من أجل تجنب أي تعارض مع النهج القائم على الزناد، والقاعدة.

وأخيرا، نضيف البيانات إلى الجدول الرئيسي، ويمكننا أن نرى في الخطوة ٤ من قسم كيف ينجز ذلك... استنادا إلى النهج القائم على القاعدة، يذهب الإدخال المقابل إلى قسم country_log_default.

تثبيت PL/Proxy

إن PL/Proxy أو وكيل لغة PL هو نظام تقسيم قاعدة بيانات ينفذ كلغة PL. يجعل PL/Proxy من السهل تقسيم جداول مستقلة كبيرة بين عقد متعددة بطريقة تسمح تقريبا بقابلية غير محدود. تعمل قابلية التوسع PL/Proxy على كل من أعباء العمل المتعلقة بالقراءة والكتابة. والفكرة الرئيسية هي عمل وظيفة الوكيل بنفس هيكلية الوظيفة التي ستستدعي عن بعد، لذلك يجب فقط أن تحدد معلومات الوجهة داخل جسم وظيفة الوكيل.

الاستعداد للعمل

هنا، سنعرض الخطوات المطلوبة لتثبيت PL/Proxy .

كيف ينجز ذلك...

قم بتنفيذ الخطوات التالية لتثبيت PL/Proxy :

١. اذهب إلى http://pgfoundry.org/projects/plproxy/ وقم بتحميل أحدث إصدار من PL/Proxy .

٢. بمجرد تحميل أحدث إصدار من PL/Proxy ، فإن الخطوة التالية هي فك أرشيف tar:

```
tar xvfz plproxy-2.5.tar.gz
```

٣. بمجرد أن يفك ملف الأرشيف، فإن الخطوة التالية هي الانتقال إلى المجلد الذي أنشئ حديثا وبدء عملية التجميع:

```
cd   plproxy-2.5
make  &&  make install
```

كيف تعمل...

تثبيت PL/Proxy هو مهمة سهلة. هنا، نقوم بتحميل المصدر من الموقع المذكور في الخطوة ١ من القسم السابق. أحدث إصدار من PL/Proxy في هذه المرحلة 2.5 i. نحن بحاجة إلى تحميل ملف tarball التي تحتوي على الإصدار ٢.٥ من PL/Proxy، وبمجرد أن تنزل، نحن بحاجة إلى القيام بالتجميع والبناء. وبهذا يكتمل تركيب PL/Proxy.

يمكنك أيضا تثبيت PL/Proxy من الحزم الثنائية، إذا كانت الحزم المثبتة مسبقا متوفرة لنظام التشغيل الخاص بك.

التقسيم مع PL/Proxy

في هذه الوصفة، سنقوم بتغطية التقسيم الأفقي باستخدام PL/Proxy.

الاستعداد للعمل

يحتاج PL/Proxy إلى أن يكون مثبتا على الجهاز المضيف. ارجع إلى الوصفة السابقة لمزيد من التفاصيل حول كيفية تثبيت PL/Proxy.

كيف ينجز ذلك...

قم بتنفيذ التسلسل التالي من الخطوات لأداء التقسيم الأفقي باستخدام PL/Proxy:

١. قم بإنشاء ثلاث قواعد بيانات جديدة، وهذا هو قاعدة بيانات وكيل واحد اسمه nodes وقاعدتي بيانات مقسمة اسمهما nodes_0000 و nodes_0001، على التوالي:

```
postgres=# create database nodes;
postgres=# create database nodes_0000;
postgres=# create database nodes_0001;
```

٢. بعد إنشاء قواعد البيانات هذه، تتمثل الخطوة التالية في إنشاء إضافة PLPROXY:

```
psql  -d  nodes
nodes=# create extension plproxy;
```

٣. الخطوة التالية هي إنشاء مخطط PLPROXY في عقد قاعدة بيانات الوكيل:

```
nodes=# create schema plproxy;
```

٤. بعد ذلك، قم بتنفيذ الملف التالي، plproxy--2.5.0.sql على عقد قاعدة بيانات الوكيل:

```
cd  /usr/pgsql-9.3/share/extension
 psql -f plproxy--2.5.0.sql  nodes

CREATE FUNCTION
CREATE LANGUAGE
CREATE FUNCTION
CREATE FOREIGN DATA WRAPPER
```

٥. ثم، قم بضبط pl/proxy باستخدام وظائف التضبيط على العقد قاعدة بيانات الوكيل:

```
psql -d   nodes
CREATE OR REPLACE FUNCTION plproxy.get_cluster_version(cluster_name text)
RETURNS int AS $$
BEGIN
    IF cluster_name = 'nodes' THEN
        RETURN 1;
    END IF;
END;
$$ LANGUAGE plpgsql;
CREATE OR REPLACE FUNCTION plproxy.get_cluster_partitions(cluster_name text)
RETURNS SETOF text AS $$
BEGIN
    IF cluster_name = 'nodes' THEN
        RETURN NEXT 'host=127.0.0.1 dbname=nodes_0000';
        RETURN NEXT 'host=127.0.0.1 dbname=nodes_0001';
        RETURN;
    END IF;
    RAISE EXCEPTION 'no such cluster: %', cluster_name;
END;
$$ LANGUAGE plpgsql SECURITY DEFINER;
CREATE OR REPLACE FUNCTION plproxy.get_cluster_config (cluster_name text, out
key text, out val text)
RETURNS SETOF record AS $$
BEGIN
    RETURN;
END;
$$ LANGUAGE plpgsql;
```

٦. بعد ذلك، قم بتسجيل الدخول إلى قواعد البيانات المقسمة وقم بإنشاء جدول المستخدمين في كل من هذه:

```
psql -d nodes_0000
nodes_0000=# CREATE TABLE users (username text PRIMARY KEY);
psql -d  nodes_0001
```

```
nodes_0001=# CREATE TABLE users (username text PRIMARY KEY);
```

٧. الآن، قم بإنشاء الدالة التالية، () enter_user ، والتي ستستخدم لإدراج أسماء المستخدمين في جدول المستخدمين:

```
psql -d   nodes_0000
CREATE OR REPLACE FUNCTION insert_user(i_username text) RETURNS text AS $$
BEGIN
    PERFORM 1 FROM users WHERE username = i_username;
    IF NOT FOUND THEN
        INSERT INTO users (username) VALUES (i_username);
        RETURN 'user created';
    ELSE
        RETURN 'user already exists';
    END IF;
END;
$$ LANGUAGE plpgsql SECURITY DEFINER;
psql  -d  nodes_0001
CREATE OR REPLACE FUNCTION insert_user(i_username text) RETURNS text AS $$
BEGIN
    PERFORM 1 FROM users WHERE username = i_username;
    IF NOT FOUND THEN
        INSERT INTO users (username) VALUES (i_username);
        RETURN 'user created';
    ELSE
        RETURN 'user already exists';
    END IF;
END;
$$ LANGUAGE plpgsql SECURITY DEFINER;
```

٨. بعد ذلك، قم بإنشاء دالة بروكسي تسمى () enter_user على عقد قاعدة بيانات البروكسي:

```
psql -d nodes
CREATE OR REPLACE FUNCTION insert_user(i_username text) RETURNS TEXT AS $$
    CLUSTER 'nodes'; RUN ON hashtext(i_username);
$$ LANGUAGE plproxy;
```

٩. تحقق من ملف pg_hba.conf؛ سوف تحتاج إلى تعيين المصادقة على الثقة، كما هو موضح هنا، ثم قم بإعادة تشغيل خدمة بوستجريسكل:

```
host    all             all             127.0.0.1/32            trust
```

١٠. الخطوة التالية ستكون ملء الأقسام عن طريق تنفيذ الاستعلام التالي على الوكيل

```
nodes=#SELECT insert_user('user_number_'||generate_series::text) FROM
generate_series(1,10000);
```

١١ - بمجرد إدخال البيانات، تحقق من السجلات المناظرة في قواعد البيانات المقسمة، nodes_0000 و nodes_0001، على النحو التالي:

```
nodes_0000=# select count(*) from users;
 count
-------
  5106
(1 row)
nodes_0001=# select count(*) from users;
 count
-------
  4894
(1 row)
```

كيف تعمل...

وفيما يلي شرح لخرج الشفرة السابقة:

- في البداية، نقوم بإنشاء ثلاث قواعد بيانات واحدة كقاعدة بيانات بروكسي اسمها nodes وقاعدة بيانات أخرى اسمها nodes_0000 و nodes_0001 وستقسم البيانات بينهما.

- بعد تنفيذ الخطوة السابقة، ستكون الخطوة التالية هي إنشاء الإضافة plproxy.

- كما هو موضح في الخطوة ٥ من القسم السابق، نقوم بتضبيط pl/proxy باستخدام وظائف التضبيط على عقد قاعدة بيانات الوكيل. تستدعى الدالة plproxy.get_cluster_partitions() عند إعادة توجيه استعلام إلى قاعدة بيانات بعيدة، وتستخدم بواسطة PL/Proxy للحصول على سلسلة الاتصال التي ستستخدم لكل قسم. يمكن أيضا استخدام الدالة () plproxy.get_cluster_version ، التي تستدعى عند كل طلب، وتستخدم لتحديد ما إذا كان الناتج من النتيجة المخزنة مؤقتا من plproxy.get_cluster_partitions يمكن إعادة استخدامها. ويمكن أيضا استخدام plproxy.get_cluster_config()، والتي تمكننا من تضبيط سلوك PL/Proxy.

- وبمجرد الانتهاء من تحديد وظائف التضبيط على عقد قاعدة بيانات الوكيل، فإن الخطوة التالية هي إنشاء مستخدمي الجدول في كل من قواعد البيانات المقسمة، التي ستقسم البيانات بينها.

- ثم، أنشأنا () function_user الدالة التي ستستخدم لإدراج أسماء المستخدمين في جدول المستخدمين. ستعرف الدالة ()enter_user على كل من قواعد البيانات المقسمة، nodes_0000 و nodes_0001. ويظهر ذلك في الخطوة ٧ من القسم السابق.

- في الخطوة التالية، نقوم بإنشاء وظيفة بروكسي، ()enter_user، داخل عقد قاعدة بيانات الوكيل. تستخدم دالة البروكسي لإرسال النتيجة INSERT إلى القسم المناسب. ويظهر ذلك في الخطوة ٨ من القسم السابق.

- أخيرا، سنقوم بملء الأقسام بالبيانات العشوائية عن طريق تنفيذ الدالة بروكسي () enter_user في عقد قواعد البيانات البروكسي المسماة. ويظهر ذلك في الخطوة ١٠ من القسم السابق.

هناك المزيد

لمزيد من التفاصيل حول كيفية استخدام PL/Proxy من أجل استعلامات الوكيل عبر مجموعة من قواعد البيانات عن بعد، راجع http://plproxy.projects.pgfoundry.org/doc/tutorial.html.

١٠ - الوصول إلى بوستجريسكل من بيرل

في هذا الفصل، سنغطي الوصفات التالية:

- إجراء اتصال إلى قاعدة بيانات بوستجريسكل باستخدام بيرل
- إنشاء الجداول باستخدام بيرل
- إدراج السجلات باستخدام بيرل
- الوصول إلى البيانات باستخدام بيرل
- تحديث السجلات باستخدام بيرل
- حذف السجلات باستخدام بيرل

مقدمة

بيرل Perl هي لغة البرمجة العامة، عالية المستوى من حيث التفسير، والديناميكية.

وبشكل عام فإن التواصل مع بوستجريسكل ينطوي على سلسلة طويلة من التلاعبات، وهذا ما تتفوق فيه بيرل كلغة.

في بيرل، تنفذ واجهات قاعدة البيانات من قبل وحدات بيرل DBI. وحدة DBI تقدم واجهة قاعدة بيانات مستقلة لتطبيقات بيرل. من ناحية أخرى، وحدة تحكم قاعدة البيانات تتعامل مع تفاصيل الوصول إلى قواعد بيانات مختلفة.

هناك ثلاث طرق للوصول إلى بوستجريسكل من بيرل، على النحو التالي:

- الوصول على مستوى منخفض، والذي يحدث عن طريق ربط بيرل بواجهة libpq C.
- الوصول رفيع المستوى، مع مساعدة من طبقة مستقلة لقاعدة البيانات.
- الوصول عن طريق تضمين مترجم بيرل.

إجراء اتصال إلى قاعدة بيانات بوستجريسكل باستخدام بيرل

سنقوم بإجراء اتصالات إلى قاعدة بيانات بوستجريسكل باستخدام بيرل.

الاستعداد للعمل

نفذت التعليمات التالية على جهاز سينتوس لينوكس، ويفترض أن لغة بيرل مثبتة مسبقا.

ويمكن الوصول إلى قاعدة بيانات بوستجريسكل باستخدام وحدة بيرل _DBI، وهي وحدة الوصول إلى قاعدة البيانات للغة البرمجة بيرل. تحدد وحدة بيرل DBI بمجموعة من الأساليب والمتغيرات والاتفاقيات التي توفر واجهة قاعدة بيانات قياسية.

وحدة DBI، في حد ذاتها، لا تملك القدرة على التواصل مع بوستجريسكل، فمن الضروري تثبيت الوحدة الخلفية المناسبة، والتي في هذه الحالة هي DBD::Pg.

على ريد هات، سينتوس، و Scientific Linux، فضلا عن توزيعات لينكس الأخرى القائمة على ريد هات، الحزمة التي توفر هذه الوحدة هي perl-DBD-Pg ويمكن تثبيتها على النحو التالي:

```
yum install perl-DBD-Pg
```

أما في الأنظمة المبنية على Debian ، فإن الحزمة التي توفر هذه الوحدة هي libdbd-pg-perl، ويمكن تثبيتها كما يلي على التوزيعات المستندة إلى أبونتو Ubuntu أو دبيان:

```
apt-get install libdbd-pg-perl
```

قبل أن نبدأ باستخدام واجهة بيرل بوستجريسكل، سنحتاج إلى إدخال آلية المصادقة والتحكم في الوصول التالي في ملف pg_hba.conf:

```
# IPv4 local connections:
host     all          all          127.0.0.1/32          md5
```

بمجرد إجراء هذه التغييرات، سنحتاج إلى إعادة تشغيل خادم بوستجريسكل:

```
$ pg_ctl -D  $PGDATA  restart
```

كيف ينجز ذلك...

يمكننا استخدام التعليمات البرمجية بيرل التالية لإجراء اتصال بقاعدة بيانات بوستجريسكل الموجودة، وهذه هي، قاعدة بيانات dvdrental، التي تقيم على نفس الجهاز وتستخدم منفذ ٥٤٣٢.

١. أولا، يمكن حفظ رمز بيرل التالية في ملف يسمى connect.pl:

```
#!/usr/bin/perl

use DBI;
use strict;
   my $driver   = "Pg";
my $database = "dvdrental";
my $dsn = "DBI:$driver:dbname=$database;host=127.0.0.1;port=5432";
my $userid = "postgres";
my $password = "postgres";
my $dbh = DBI->connect($dsn, $userid, $password, { RaiseError => 1 })
                      or die $DBI::errstr;

print "CONNECTION TO THE  DVDRENTAL DATABASE MADE SUCCESSFULLY\n";
```

٢. الخطوة التالية هي تغيير الأذونات، على النحو التالي:

```
chmod 755 connect.pl
```

٣. يمكن بعد ذلك تنفيذ برنامج بيرل في سطر الأوامر، على النحو التالي:

```
bash-3.2$ perl connect.pl
CONNECTION MADE TO THE  DVDRENTAL DATABASE MADE SUCCESSFULLY
```

كما يمكن أن ترى من الناتج السابق، أثناء تنفيذ البرنامج، تشير رسالة الناتج إلى أن الاتصال بقاعدة بيانات dvdrental قد نجح.

كيف تعمل...

نقوم بالاتصال بقاعدة البيانات باستخدام وظيفة connect. حيث تقوم بإرجاع مقبض اتصال المطلوب عند إجراء المكالمات إلى وحدة DBI. تتطلب وظيفة connect الوسائط التالية:

```
  connect($data_source, "userid", "password", \%attr);
```

الوسيطة الأولى لوظيفة الاتصال هي اسم مصدر البيانات، وهو كيان واحد يتكون من اسم قاعدة البيانات واسم المضيف أو عنوان IP واختياريا رقم المنفذ. يتكون مصدر البيانات أيضا من البادئة Pg، وهو برنامج تشغيل قاعدة البيانات بوستجريسكل لوحدة DBI.

الوسيطة الثانية هي userid و اسم المستخدم الذي يجرى اتصال إلى قاعدة بيانات بوستجريسكل عن طريقه.

الوسيطة الثالثة هي كلمة المرور، وهي كلمة مرور المستخدم الذي يبدأ اتصال قاعدة البيانات. إذا ما حددت سلسلة فارغة لكلمة السر، سيقوم بيرل بالبحث عن قيمة كلمة السر في متغيرات البيئة، DBI_USER و DBI_PASS، مما قد يؤدي إلى فشل التعليمات البرمجية أثناء تنفيذها. لذلك، نحن بحاجة إلى توخي الحذر في مثل هذه السيناريوهات.

الوسيطة الأخيرة اختيارية، وهي تشير إلى أي سمات يمكن استخدامها.

في التعليمات البرمجية السابقة، في المقطع كيف ينجز ذلك...، استخدمنا دالة الاتصال، كما يلي:

```
my $dbh = DBI->connect($dsn, $userid, $password, { RaiseError => 1 })
```

الوسيطة الأولى المستخدمة هنا هي متغير dsn، الذي عرف في البداية، على النحو التالي:

```
my $dsn = "DBI:$driver:dbname=$database;host=127.0.0.1;port=5432";
```

هنا، يمكننا أن نرى أن قيمة dsn هو كيان واحد، والذي يتألف من سائق، وهو سائق Pg، و dbname الذي يعرف في متغير قاعدة البيانات وقيمته هي dvdrental؛ ويتكون كذلك من اسم المضيف ورقم المنفذ الذي عرف صراحة هنا.

الوسيطة الثانية هي متغير userid الذي يستخدم قيمة postgres المعرفة مسبقا في التعليمات البرمجية.

الوسيطة الثالثة هي متغير كلمة السر، الذي قيمته postgres.

الوسيطة الأخيرة المستخدمة هي السمة RaiseError التي تسببها وحدة واجهة قاعدة البيانات و بالتالي استدعاء حالة HandleError أو موت الاتصال إذا لم تعرف حالة HandleError عند اكتشاف خطأ قاعدة بيانات.

> إذا كنت بحاجة إلى مزيد من المعلومات حول وظيفة الاتصال، يمكنك استخدام الروابط التالية للحصول على شرح أكثر تفصيلا:
>
> - http://oreilly.com/catalog/perldbi/chapter/ch04.html
> - http://search.cpan.org/~rudy/DBD-Pg-1.32/Pg.pm

إنشاء الجداول باستخدام بيرل

في هذه الوصفة، سوف نعرض لك كيفية إنشاء الجداول في قاعدة بيانات بوستجريسكل باستخدام بيرل.

الاستعداد للعمل

سنستخدم المشغل qq، وستحتوي الإعدادات التي ستمرر إلى المشغل على عبارة SQL: CREATE TABLE. يستخدم المشغل qq لإرجاع سلسلة مقتبسة مزدوجة. قبل إنشاء الجدول، يجب علينا أولا استخدام دالة connect للاتصال بقاعدة بيانات بوستجريسكل.

كيف ينجز ذلك...

يمكننا استخدام التعليمات البرمجية التالية لإنشاء جدول بالاسم EMPLOYES. سيخزن هذا الجدول في قاعدة بيانات dvdrental لأن الاتصال الذي قام به محول بوستجريسكل هو قاعدة البيانات dvdrental. قم بحفظ التعليمات البرمجية التالية في ملف يسمى createtable.pl، والذي سيشغل لاحقا:

```
#!/usr/bin/perl

use DBI;
use strict;

my $driver   = "Pg";
my $database = "dvdrental";
my $dsn = "DBI:$driver:dbname=$database;host=127.0.0.1;port=5432";
my $userid = "postgres";
my $password = "postgres";
my $dbh = DBI->connect($dsn, $userid, $password, { RaiseError => 1 })
                      or die $DBI::errstr;
print "Dvdrental database opened \n";

my $stmt  = qq(CREATE TABLE EMPLOYEES
      (ID INT PRIMARY KEY     NOT NULL,
       NAME             TEXT    NOT NULL,
       AGE              INT     NOT NULL,
       ADDRESS          CHAR(60),
       SALARY           REAL););
my $rv = $dbh->do($stmt);
 print "EMPLOYEES table created successfully\n\n";

$dbh->disconnect();
```

```
bash-3.2$ perl createtable.pl

Dvdrental database opened
EMPLOYEES table created successfully
```

في الناتج السابق، يمكنك أن ترى أنه عند تنفيذ الملف الذي يحتوي على التعليمات البرمجية السابقة أجري اتصال إلى قاعدة بيانات dvdrental وأنشئ جدول EMPLOYEES. يمكن مشاهدة ذلك من رسالة سطر الأوامر، "EMPLOYEES table created successfully"

كيف تعمل...

من وجهة نظر إنشاء الجدول، فإن الجزء التالي من التعليمات البرمجية السابقة الذي يحتاج إلى تفسير:

```
my $rv = $dbh->do($stmt);
```

هنا، نحن نستخدم المعالج الذي المرجع بواسطة الدالة connect بالاقتران مع الدالة القيام بتنفيذ عبارة CREATE_TABLE مررت في متغير $stmt كما يلي:

```
my $stmt = qq(CREATE TABLE EMPLOYEES
      (ID INT PRIMARY KEY     NOT NULL,
       NAME           TEXT    NOT NULL,
       AGE            INT     NOT NULL,
       ADDRESS        CHAR(60),
       SALARY         REAL););
```

طريقة () Do عبارة عن مزيج من ()prepare و () execute. يمكن استخدامها فقط لعبارات أخرى غير SELECT، حيث لا تحتاج إلى مقبض العبارة للوصول إلى نتائج الاستعلام. ترجع () Do عدد الصفوف المتأثرة.

يتم استخدام طريقة () disconnect في القسم السابق لإنهاء جلسة قاعدة البيانات الحالية وقطع الاتصال من قاعدة البيانات.



لمزيد من التفاصيل حول الجداول في بيرل، راجع

http://www.postgresql.org/docs/9.3/interactive/plperl-builtins.html

إدراج السجلات باستخدام بيرل

في هذه الوصفة، سنقوم بإدراج سجلات جديدة في الجدول EMPLOYEES في قاعدة البيانات dvdrental.

الاستعداد للعمل

قبل إدراج السجلات في الجدول، نحتاج أولا إلى استخدام الدالة connect، من أجل الاتصال بقاعدة البيانات. نوقشت وظيفة connect في الوصفة الأولى من الفصل.

كيف ينجز ذلك...

سنقوم باستخدام التعليمات البرمجية التالية لإدراج سجلات جديدة في الجدول EMPLOYEES:

```
#!/usr/bin/perl

use DBI;
use strict;

my $driver   = "Pg";
my $database = "dvdrental";
my $dsn = "DBI:$driver:dbname=$database;host=127.0.0.1;port=5432";
my $userid = "postgres";
my $password = "postgres";
my $irows = 0;
my $dbh = DBI->connect($dsn, $userid, $password, { RaiseError => 1 })
                    or die $DBI::errstr;
print "Opened database successfully\n";

my $stmt = qq(INSERT INTO EMPLOYEES (ID,NAME,AGE,ADDRESS,SALARY)
    VALUES (5, 'SandeepSingh', 37, 'Saharanpur', 90000.00 ));
my $rv = $dbh->do($stmt);


$irows = $rv + $irows;

$stmt = qq(INSERT INTO EMPLOYEES (ID,NAME,AGE,ADDRESS,SALARY)
    VALUES (6, 'AmitGovil', 37, 'Aligarh', 85000.00 ));
$rv = $dbh->do($stmt);


$irows = $rv + $irows;

$stmt = qq(INSERT INTO EMPLOYEES (ID,NAME,AGE,ADDRESS,SALARY)
    VALUES (7, 'NeerajKumar', 38, 'Rohtak', 90000.00 ));
$rv = $dbh->do($stmt);
```

```
$irows = $rv + $irows;

$stmt = qq(INSERT INTO EMPLOYEES (ID,NAME,AGE,ADDRESS,SALARY)
    VALUES (8, 'SandeepSharma', 36, 'Gurgaon ', 75000.00 ););
$rv = $dbh->do($stmt);


$irows = $rv + $irows;

print "Number of rows inserted : $irows\n";
print "New Records created successfully\n";
$dbh->disconnect();
```

تحفظ التعليمات البرمجية السابقة في ملف يسمى enter.pl، ونحن نحصل على خرج سطر الأوامر التالي عندما تدرج السجلات الجديدة بنجاح:

```
bash-3.2$ perl  insert.pl
Opened database successfully
New Records created successfully
```

كيف تعمل...

للحصول على شرح التعليمات البرمجية السابقة، نأخذ جزء من التعليمات البرمجية التي سوف تظهر كيفية إدراج السجلات في الجدول EMPLOYEES:

```
$stmt = qq(INSERT INTO EMPLOYEES (ID,NAME,AGE,ADDRESS,SALARY)
    VALUES (7, 'NeerajKumar', 38, 'Rohtak', 90000.00 ));
$rv = $dbh->do($stmt);
```

إذا قمت بإلقاء نظرة على التعليمات البرمجية السابقة، يمكنك أن ترى أن أولا، استخدمنا عبارة INSERT، والسجلات المعرفة، وتمرير عبارة INSERT SQL إلى متغير. بعد القيام بذلك، استخدمنا الدالة do() لإرجاع نتيجة عبارة INSERT في الجدول. ثم نفذت نفس الخطوات بالتتابع لبيانات INSERT الأخرى المستخدمة في التعليمات البرمجية.

في القسم السابق، نستخدم أيضا متغير irows لتتبع عدد الصفوف التي أدرجت في الجدول. في كل مرة نقوم بإدراج سجل في الجدول، وضعنا الشرط كما هو مبين في السطر التالي من التعليمات البرمجية. في البداية، تعين قيمة متغير irows إلى الصفر وعندما نقوم بإدراج سجل في جدول PLPLOYEES، تعين قيمة متغير rv إلى ١. لذلك، وفقا للشرط

التالي، في كل مرة هناك تغيير أو إدراج سجل، فإن قيمة المتغير irows تزيد بمقدار ١ وهلم جرا، حتى تدرج كافة السجلات وتتوقف في نهاية المطاف في ٤ للإشارة إلى أن أربع سجلات أدرجت في المجموع:

```
$irows = $rv + $irows;
```

الوصول إلى بيانات الجدول باستخدام بيرل

في هذه الوصفة، سنناقش كيفية الوصول إلى بيانات الجدول من قاعدة بيانات بوستجريسكل باستخدام بيرل.

الاستعداد للعمل

إن اتصال قاعدة البيانات يعتبر إلزاميا قبل أن نتمكن من تحديد البيانات. وبالتالي فالدالة () connect هي الأولى التي يجب استدعاءها لإجراء اتصال قاعدة بيانات قبل الوصول إلى البيانات.

كيف ينجز ذلك...

يمكننا استخدام التعليمات البرمجية التالية للوصول إلى البيانات من الجدول EMPLOYEES الموجود في قاعدة البيانات dvdrental. تحفظ التعليمات البرمجية التالية في ملف يسمى select.pl، والذي سيشغل في وقت لاحق من سطر الأوامر:

```
#!/usr/bin/perl

use DBI;
use strict;

my $driver   = "Pg";
my $database = "dvdrental;";
my $dsn = "DBI:$driver:dbname=$database;host=127.0.0.1;port=5432";
my $userid = "postgres";
my $password = "postgres";
my $dbh = DBI->connect($dsn, $userid, $password, { RaiseError => 1 })
                      or die $DBI::errstr;
print "Opened database successfully\n";

my $stmt = qq(SELECT id, name, address, salary  from EMPLOYEES;);
my $sth = $dbh->prepare($stmt) or die "Cannot prepare: " . $dbh->errstr();
```

```
my $rv = $sth->execute() or die "Cannot execute: " . $sth->errstr();

while(my @row = $sth->fetchrow_array()) {
      print "ID = ". $row[0] . "\n";
      print "NAME = ". $row[1] ."\n";
      print "ADDRESS = ". $row[2] ."\n";
      print "SALARY =  ". $row[3] ."\n\n";
}
$sth->finish();
print "select Operation done successfully\n";
$dbh->disconnect();
```

فيما يلي ناتج الرمز السابق:

```
bash-3.2$ perl select.pl
Opened database successfully
ID = 5
NAME = SandeepSingh
ADDRESS = Saharanpur
SALARY =  90000

ID = 6
NAME = AmitGovil
ADDRESS = Aligarh
SALARY =  85000

ID = 7
NAME = NeerajKumar
ADDRESS = Rohtak
SALARY =  90000

ID = 8
NAME = SandeepSharma
ADDRESS = Gurgaon
SALARY =  75000

select Operation done successfully
```

كيف تعمل...

وفيما يلي جزء من التعليمات البرمجية التي تتناول أساسا اختيار السجلات من جدول:

```
my $stmt = qq(SELECT id, name, address, salary  from EMPLOYEES;);
my $sth = $dbh->prepare($stmt) or die "Cannot prepare: " . $dbh->errstr();

my $rv = $sth->execute() or die "Cannot execute: " . $sth->errstr();

while(my @row = $sth->fetchrow_array()) {
      print "ID = ". $row[0] . "\n";
      print "NAME = ". $row[1] ."\n";
      print "ADDRESS = ". $row[2] ."\n";
      print "SALARY =  ". $row[3] ."\n\n";
}
$sth->finish();
```

في التعليمات البرمجية السابقة، أول شيء نقوم به هو كتابة الاستعلام SELECT لدينا وتمريره إلى متغير $stmt.

الخطوة التالية هي استخدام دالة () prepare من أجل إعداد عبارة SQL، والتي يمكن تنفيذها في وقت لاحق من قبل مشغل قاعدة البيانات وإرجاع مرجع إلى كائن مقبض العبارة.

الخطوة التالية هي استخدام الدالة () execute لتنفيذ عبارة SQL المعدة.

وأخيرا، من أجل جلب نتائج الأمر SELECT، نستخدم الدالة () fetchrow_array. تحصل الدالة () fetchrow_array على الصف التالي وإرجاعها كقائمة من قيم الحقول. نحن نستخدم حلقة while لمرور على كل قيم الحقول في صف معين عبر الصف المسمى row ثم الانتقال إلى الصف التالي. ثم تكرر نفس تسلسل الأحداث حتى نصل إلى آخر قيمة الحقل الأخير في الصف الأخير.

تحديث السجلات باستخدام بيرل

هنا، سنعرض كيفية تحديث السجلات الموجودة في جدول في قاعدة بيانات بوستجريسكل باستخدام بيرل.

الاستعداد للعمل

في هذه الوصفة، أولا سنقوم بعرض عدد السجلات الموجودة في الجدول. ثم، سنقوم بتحديث بعض السجلات، ورؤية عدد السجلات حدثت، ثم مشاهدة السجلات التي غيرت في الجدول عند الوصول إلى سجلات الجدول مرة أخرى.

كيف ينجز ذلك...

سنقوم في هذا القسم بتحديث السجلات الموجودة لجدول EMPLOYEES.

١. أولا، نتحقق من السجلات الموجودة في الجدول EMPLOYEES، على النحو التالي:

```
dvdrental=# select * from EMPLOYEES;
 id |      name       | age | address         | salary
----+----------------+-----+----------------+--------
  5 | SandeepSingh   |  37 | Saharanpur     |  90000
  6 | AmitGovil      |  37 | Aligarh        |  85000
  7 | NeerajKumar    |  38 | Rohtak         |  90000
  8 | SandeepSharma  |  36 | Gurgaon        |  75000
```

٢. بعد ذلك، نستخدم رمز بيرل التالي لتحديث بعض السجلات الموجودة في الجدول EMPLOYEES وحفظ التعليمات البرمجية التالية في ملف يسمى update.pl:

```
#!/usr/bin/perl

use DBI;
use strict;

my $driver   = "Pg";
my $database = "dvdrental";
my $dsn = "DBI:$driver:dbname=$database;host=127.0.0.1;port=5432";
my $userid = "postgres";
my $password = "postgres";
my $dbh = DBI->connect($dsn, $userid, $password, { RaiseError => 1 })
                      or die $DBI::errstr;
print "Opened database successfully\n";

my $stmt = qq(UPDATE EMPLOYEES set SALARY = 55000.00 where ID=5;);
my $rv = $dbh->do($stmt);


   print "Number of rows updated : $rv\n";



$stmt = qq(SELECT id, name, address, salary  from EMPLOYEES;);
my $sth = $dbh->prepare( $stmt ) or die "Check again: " . $dbh->errstr();

$rv = $sth->execute() or die "Cannot execute: " . $sth->errstr();

while(my @row = $sth->fetchrow_array()) {
      print "ID = ". $row[0] . "\n";
      print "NAME = ". $row[1] ."\n";
      print "ADDRESS = ". $row[2] ."\n";
      print "SALARY =  ". $row[3] ."\n\n";
```

```
}
$sth->finish();
print "Operation Completed successfully\n";
$dbh->disconnect();
```

في التعليمات البرمجية السابقة، نستخدم عبارة UPDATE لتعيين قيمة SALARY إلى ٥٥٠٠٠، حيث قيمة العمود ID هي ٥.

٣. بعد ذلك، سنعرض السجلات المحدثة ونرى السجلات التي غيرت مرئية في الجدول EMPLOYEES، كما هو موضح في إخراج التعليمات البرمجية التالي:

```
bash-3.2$ perl update.pl
Opened database successfully
Number of rows updated : 1
ID = 6
NAME = AmitGovil
ADDRESS = Aligarh
SALARY =  85000

ID = 7
NAME = NeerajKumar
ADDRESS = Rohtak
SALARY =  90000

ID = 8
NAME = SandeepSharma
ADDRESS = Gurgaon
SALARY =  75000

ID = 5
NAME = SandeepSingh
ADDRESS = Saharanpur
SALARY =  55000

Operation Completed successfully
```

كيف تعمل...

فيما يلي مقتطف من التعليمات البرمجية التي استخدمت لتحديث السجلات الموجودة باستخدام بيرل في الجدول EMPLOYEES:

```
my $stmt = qq(UPDATE EMPLOYEES set SALARY = 55000.00 where ID=5;);
```

```
my $rv = $dbh->do($stmt);
print "Number of rows updated : $rv\n";
```

ويتمثل الشرط الأولي الأول في الاتصال بقاعدة البيانات باستخدام الدالة () connect التي شرحت في الوصفة الأولى للفصل.

بمجرد إجراء اتصال إلى قاعدة بيانات dvdrental، نستخدم عبارة UPDATE وتمرير عبارة SQL UPDATE في متغير $stmt. بعد القيام بذلك، الخطوة التالية هي استخدام الدالة do() لإرجاع نتيجة عبارة UPDATE المتضمنة في متغير $stmt. كما نستخدم متغير يسمى $rv والذي يستخدم لتعقب عدد السجلات المحدثة، إن وجدت. بمجرد القيام بذلك، الخطوة التالية هي جلب السجلات من الجدول من أجل التحقق من صحة التغييرات التي أجريت كجزء من استخدام عبارة UPDATE.

حذف السجلات باستخدام بيرل

في هذه الوصفة، سوف نعرض لك كيفية حذف السجلات في جدول باستخدام بيرل.

الاستعداد للعمل

في هذه الوصفة، سنعرض أولا عدد السجلات الموجودة في الجدول. ثم، سنقوم بحذف بعض السجلات، ورؤية عدد السجلات المحذوفة، ثم رؤية عدد السجلات المتوفرة في الجدول بعد الحذف.

كيف ينجز ذلك...

في هذا القسم سوف نقوم بحذف السجلات الموجودة جدول EMPLOYEES.

١. أولا، سنقوم بفحص السجلات الموجودة في الجدول EMPLOYEES، كما هو موضح هنا:

```
dvdrental=# select * from EMPLOYEES;
 id |     name       | age | address           | salary
----+----------------+-----+-------------------+--------
  6 | AmitGovil      |  37 | Aligarh           |  85000
  7 | NeerajKumar    |  38 | Rohtak            |  90000
  8 | SandeepSharma  |  36 | Gurgaon           |  75000
  5 | SandeepSingh   |  37 | Saharanpur        |  55000
(4 rows)
```

٢. بعد ذلك، سنستخدم كود بيرل التالي لحذف بعض السجلات من الجدول EMPLOYEES وحفظ الشفرة في ملف يسمى delete.pl، والذي سنقوم بتنفيذه من سطر الأوامر:

```
#!/usr/bin/perl

use DBI;
use strict;

my $driver   = "Pg";
my $database = "dvdrental";
my $dsn = "DBI:$driver:dbname=$database;host=127.0.0.1;port=5432";
my $userid = "postgres";
my $password = "postgres";
my $dbh = DBI->connect($dsn, $userid, $password, { RaiseError => 1 })
                      or die $DBI::errstr;
print "Opened database successfully\n";

my $stmt = qq(DELETE from EMPLOYEES where ID=6;);
my $rv = $dbh->do($stmt);

   print "Number of rows deleted : $rv\n";

$stmt = qq(SELECT id, name, address, salary  from EMPLOYEES;);
my $sth = $dbh->prepare( $stmt ) or die "Cannot prepare: " . $dbh->errstr();

$rv = $sth->execute() or die "Cannot execute: " . $sth->errstr();


while(my @row = $sth->fetchrow_array()) {
      print "ID = ". $row[0] . "\n";
      print "NAME = ". $row[1] ."\n";
      print "ADDRESS = ". $row[2] ."\n";
      print "SALARY =  ". $row[3] ."\n\n";
}
$sth->finish();
print "Operation done successfully\n";
$dbh->disconnect();
```

هنا، في التعليمات البرمجية السابقة، تستخدم عبارة DELETE التي نقوم بإصدارها لحذف سجل من الجدول حيث قيمة العمود ID هو ٦.

٣. بعد ذلك تحقق من إخراج سطر الأوامر من التعليمات البرمجية السابقة:

```
bash-3.2$ perl delete.pl
Opened database successfully
Total number of rows deleted : 1
ID = 7
NAME = NeerajKumar
ADDRESS = Rohtak
SALARY =  90000

ID = 8
NAME = SandeepSharma
ADDRESS = Gurgaon
SALARY =  75000

ID = 5
NAME = SandeepSingh
ADDRESS = Saharanpur
SALARY =  55000

Operation done successfully
```

كيف تعمل...

المقتطف التالي من التعليمات البرمجية التي استخدمت لحذف السجلات من الجدول -PLPLOYEES:

```
my $stmt = qq(DELETE from EMPLOYEES where ID=6;);
my $rv = $dbh->do($stmt);

   print "Number of rows deleted : $rv\n";
```

الشرط الأولي الأول هو الاتصال بقاعدة البيانات باستخدام الدالة ()connect التي شرحناها في الوصفة الأولى من الفصل.

عندما يتم الاتصال إلى قاعدة بيانات dvdrental، نستخدم عبارة DELETE وتمرير العبارة DELETE SQL في متغير $stmt. بعد القيام بذلك، الخطوة التالية هي الاستخدام الدالة ()do من أجل إرجاع نتيجة العبارة DELETE المضمنة في المتغير $stmt. نستخدم أيضا متغير يسمى $rv، والذي يستخدم لتعقب عدد الصفوف المحذوفة من الجدول. بمجرد القيام بذلك، الخطوة التالية هي جلب السجلات من الجدول لمشاهدة قائمة السجلات المتوفرة في الجدول EMPLOYEES.

١١ - الوصول إلى بوستجريسكل من بايثون

في هذا الفصل، سنغطي الوصفات التالية:

- إجراء اتصالات إلى قاعدة بيانات بوستجريسكل باستخدام بايثون
- إنشاء الجداول باستخدام بايثون
- إدراج السجلات باستخدام بايثون
- الوصول إلى البيانات باستخدام بايثون
- تحديث السجلات باستخدام بايثون
- حذف السجلات باستخدام بايثون

مقدمة

بايثون لغة برمجة عالية المستوى، كائنية التوجه، وذات أغراض عامة.

بايثون لغة برمجة مفتوحة المصدر، مصممة تصميما جيدا، وتعتبر لغة قوية ومحمولة. تملك بايثون قواعد لغوية سهلة التعلم، ومع المميزات البرمجية المتقدمة فإنها تستخدم على نطاق واسع من قبل المطورين والإداريين في جميع أنحاء العالم.توفر بايثون وسيلة سهلة للوصول إلى قاعدة البيانات عبر DB API الذي يوفر الحد الأدنى من المعايير للعمل مع قواعد البيانات باستخدام بناء الجملة في بايثون ودلالاتها.

والخطوات المستخدمة في استخدام دوال بايثون تكون على النحو التالي:

- استيراد وحدة -API
- إنشاء جلسة قاعدة بيانات
- تنفيذ بيانات SQL
- إغلاق جلسة قاعدة البيانات

إجراء اتصالات إلى قاعدة بيانات بوستجريسكل باستخدام بايثون

هنا، في هذه الوصفة، سنقوم بإجراء اتصالات إلى قاعدة بيانات بوستجريسكل باستخدام لغة بايثون.

الاستعداد للعمل

نفذت التعليمات التالية على جهاز سينتوس لينكس ويفترض أن لغة بايثون مثبتة بالفعل.

يمكن الوصول إلي قاعدة بيانات بوستجريسكل باستخدام وحدة psycopg2 وهو محول قاعدة بيانات للغة بايثون.

يمكن تثبيت هذا، على النحو التالي، على جهاز سينتوس:

```
sudo yum install python-psycopg2
```

كيف ينجز ذلك...

يمكننا استخدام التعليمات البرمجية بايثون التالية لإجراء اتصالات لقاعدة بيانات بوستجريسكل موجودة، وهي قاعدة بيانات dvdrental التي تقع على نفس الجهاز وتستخدم منفذ ٥٤٣٢.

يمكن حفظ كود بايثون التالي في ملف يسمى connect.py:

```
#!/usr/bin/python

import psycopg2

conn = psycopg2.connect(database="dvdrental", user="postgres",
password="postgres", host="127.0.0.1", port="5432")

print "Opened DVD Rental Database Successfully Using Python"
```

ويمكن بعد ذلك تنفيذ رمز بايثون السابق في سطر الأوامر، كما يلي:

```
-bash-3.2$ python connect.py
```

فيما يلي ناتج الرمز السابق:

```
Opened DVD Rental Database Successfully Using Python
```

كما يمكن أن نلاحظ من الناتج السابق، أنه بعد تشغيل البرنامج فإن رسالة الإخراج تظهر أن الاتصال إلى قاعدة البيانات dvdrental قد نجح.

تستخدم الدالة connect للاتصال بقاعدة البيانات التي تقوم بإرجاع كائن اتصال. تتكون دالة connect المستخدمة هنا في التعليمات البرمجية السابقة من المعلمات التالية:

```
connect("db", "userid", "password", host,port);
```

الوسيطة الأولى لوظيفة الاتصال هي قاعدة البيانات التي يراد الاتصال بها.

الوسيطة الثانية هي userid أو اسم المستخدم الذي يجري اتصال إلى قاعدة بيانات بوستجريسكل.

الوسيطة الثالثة هي كلمة المرور التي هي كلمة مرور المستخدم الذي يبدأ اتصال قاعدة البيانات.

الوسيطة الرابعة تشير إلى اسم المضيف أو عنوان IP الخاص بالخادم الذي يستضيف قاعدة البيانات.

تشير الوسيطة التالية إلى رقم المنفذ لقاعدة البيانات التي يمكن للعميل بدء اتصال قاعدة البيانات.

في التعليمات البرمجية السابقة، في المقطع كيف ينجز ذلك...، استخدمنا دالة الاتصال، كما يلي:

```
conn = psycopg2.connect(database="dvdrental", user="postgres",
password="postgres", host="127.0.0.1", port="5432")
```

هنا، نحن قمنا بالاتصال بقاعدة بيانات dvdrental عن طريق المستخدم postgres وكلمة المرور postgres. الجهاز الذي نربطه هو المضيف المحلي وخادم قاعدة البيانات يستمع لمنفذ ٥٤٣٢ للاتصالات.

إذا كنت بحاجة إلى مزيد من المعلومات حول وظيفة connect يمكنك استخدام الروابط التالية للحصول على شرح أكثر تفصيلا:

```
http://blogs.wrox.com/article/using-the-python-database-apis/
http://initd.org/psycopg/docs/module.html#psycopg2.connect
```

إنشاء جداول باستخدام بايثون

هنا، في هذه الوصفة، سنناقش كيفية إنشاء الجداول في قاعدة بيانات بوستجريسكل باستخدام لغة بايثون.

الاستعداد للعمل

قبل إنشاء جدول، نحتاج أولا إلى إجراء اتصال بقاعدة بيانات بوستجريسكل باستخدام الدالة الاتصال، وبمجرد فتح قاعدة البيانات، فإنه يمكننا استخدام عبارات DDL لإنشاء الجدول.

كيف ينجز ذلك...

يمكننا استخدام التعليمات البرمجية التالية لإنشاء جدول باسم EMPLOYEES.سيخزن هذا الجدول في قاعدة بيانات dvdrental لأن الاتصال الذي قام به محول بوستجريسكل كان إلى قاعدة بيانات dvdrental. يمكن حفظ الشفرة التالية في ملف يسمى createtable.py والذي سنشغله لاحقا:

```
#!/usr/bin/python

import psycopg2

conn = psycopg2.connect(database="dvdrental", user="postgres",
password="postgres", host="127.0.0.1", port="5432")
print "Opened DVD Rental Database Successfully"

cur = conn.cursor()
cur.execute('''CREATE TABLE EMPLOYEES
       (ID INT PRIMARY KEY     NOT NULL,
       NAME           TEXT    NOT NULL,
       AGE            INT     NOT NULL,
       ADDRESS        CHAR(50),
       SALARY         REAL);''')
print "Table created successfully"

conn.commit()
conn.close()
```

في الخرج التالي، يمكننا أن نرى أنه عند تنفيذ ملف createtable.py الذي يحتوي على التعليمات البرمجية السابقة، فإنه يجرى اتصال إلى قاعدة بيانات dvdrental وينشئ الجدول EMPLOYEES. يمكن رؤية ذلك من رسالة سطر الأوامر "Table created successfully":

```
bash-3.2$ python createtable.py
Opened DVD Rental Database Successfully
Table created successfully
```

كيف تعمل...

من وجهة نظر إنشاء الجدول، فإن الجزء التالي من التعليمات البرمجية السابقة هو الذي يحتاج إلى تفسير:

```
cur = conn.cursor()
cur.execute('''CREATE TABLE EMPLOYEES
       (ID INT PRIMARY KEY     NOT NULL,
       NAME           TEXT    NOT NULL,
       AGE            INT     NOT NULL,
       ADDRESS        CHAR(50),
       SALARY         REAL);''')
print "Table created successfully"

conn.commit()
conn.close()
```

أولا نقوم بإنشاء كائن المؤشر عن طريق استدعاء وظيفة مؤشر ()cursor. وعندما يحدث ذلك، نستخدم الدالة كائن المؤشر ()execute لتنفيذ عبارة CREATE TABLE لإنشاء جدول. ثم، نستدعي دالة commit لحفظ التغييرات، وأخيرا نستدعي الدالة ()close لإغلاق اتصال قاعدة البيانات.

إدراج السجلات باستخدام بايثون

في هذه الوصفة سنقوم بإدراج سجلات جديدة في الجدول EMPLOYEES في قاعدة بيانات dvdrental.

الاستعداد للعمل

قبل إدراج السجلات في الجدول، نحتاج أولا إلى استخدام دالة connect للاتصال بقاعدة البيانات أولا. لقد قمنا بمناقشة وظيفة الاتصال في الوصفة الأولى من هذا الفصل.

كيف ينجز ذلك...

سنقوم باستخدام التعليمات البرمجية التالية لإدراج سجلات جديدة في الجدول EMPLOYEES:

```
#!/usr/bin/python

import psycopg2

conn = psycopg2.connect(database="dvdrental", user="postgres",
password="postgres", host="127.0.0.1", port="5432")
print "Opened database successfully"
```

```
cur = conn.cursor()

cur.execute("INSERT INTO EMPLOYEES (ID,NAME,AGE,ADDRESS,SALARY) \
    VALUES (1, 'SandeepSingh', 39, 'Saharanpur', 90000.00 )");

cur.execute("INSERT INTO EMPLOYEES (ID,NAME,AGE,ADDRESS,SALARY) \
    VALUES (2, 'NeerajKumar', 42, 'Rohtak', 90000.00 )");

cur.execute("INSERT INTO EMPLOYEES (ID,NAME,AGE,ADDRESS,SALARY) \
    VALUES (3, 'AmitGovil', 37, 'Aligarh', 88000.00 )");

cur.execute("INSERT INTO EMPLOYEES (ID,NAME,AGE,ADDRESS,SALARY) \
    VALUES (4, 'SandeepSharma', 36, 'Haridwar ', 75000.00 )");

conn.commit()
print "Records created successfully in EMPLOYEES Table";
conn.close()
```

احفظ النص البرمجي السابق في ملف يسمى insert.py وفيما يلي خرج سطر الأوامر الذي نحصل عليه بمجرد إدراج السجلات الجديدة بنجاح:

```
bash-3.2$ python insert.py
Opened database successfully
Records created successfully in Employees Table
```

كيف تعمل...

لشرح النص البرمجي السابق، نأخذ مقتطف من التعليمات البرمجية التي سوف تظهر كيفية عمل إدراج السجلات في الجدول EMPLOYEES.

```
cur = conn.cursor()

cur.execute("INSERT INTO EMPLOYEES (ID,NAME,AGE,ADDRESS,SALARY) \
    VALUES (1, 'SandeepSingh', 39, 'Saharanpur', 90000.00 )");
```

إذا رأينا التعليمات البرمجية السابقة أولا، ثم يمكننا أن نلاحظ أنه بمجرد إجراء الاتصال إلى قاعدة البيانات، نستخدم الدالة كائن اتصال الدالة () cursor الذي يخلق كائن المؤشر الذي سنستخدمه في تنفيذ عبارات SQL منها. وهنا، في التعليمات البرمجية السابقة، يمكننا أن نرى أن كائن المؤشر خزن في متغير cur ثم نسمي الدالة كائن المؤشر ()execute

لتنفيذ عبارات INSERT وهكذا لعبارات SQL أخرى. في نهاية المطاف نستدعي commit() للتأكد من أن التغييرات التي أجريت / السجلات المدرجة حفظت في قاعدة البيانات.

الوصول إلى بيانات الجدول باستخدام بايثون

في هذه الوصفة، سنرى كيفية الوصول إلى بيانات الجدول من قاعدة بيانات بوستجريسكل باستخدام بايثون.

الاستعداد للعمل

إن اتصال قاعدة البيانات إلزامي قبل أن نتمكن من تحديد البيانات. ولهذا السبب تعتبر الدالة () connect هي الدالة الأولى التي يجب استدعاءها أولا لإجراء اتصال قاعدة بيانات قبل الوصول إلى البيانات.

كيف ينجز ذلك...

يمكننا استخدام التعليمات البرمجية التالية للوصول إلى البيانات من الجدول EMPLOYEES موجودة في قاعدة بيانات dvdrental. احفظ الشفرة التالية في ملف يسمى select.py ، والذي سنشغله لاحقا من سطر الأوامر:

```
#!/usr/bin/python

import psycopg2

conn = psycopg2.connect(database='dvdrental', user='postgres',
                        password='postgres', host='127.0.0.1',
                        port='5432')
print 'Opened database successfully'

cur = conn.cursor()

cur.execute('SELECT id, name, address, salary  from EMPLOYEES')
rows = cur.fetchall()
for row in rows:
    print 'ID = ', row[0]
    print 'NAME = ', row[1]
    print 'ADDRESS = ', row[2]
    print 'SALARY = ', row[3], '\n'

print 'Select Operation done successfully'
```

```
conn.close()
```

فيما يلي ناتج الشفرة السابق:

```
bash-3.2$ python select.py
Opened database successfully
ID =  1
NAME =  SandeepSingh
ADDRESS =  Saharanpur
SALARY =  90000.0

ID =  2
NAME =  NeerajKumar
ADDRESS =  Rohtak
SALARY =  90000.0

ID =  3
NAME =  AmitGovil
ADDRESS =  Aligarh
SALARY =  88000.0

ID =  4
NAME =  SandeepSharma
ADDRESS =  Haridwar
SALARY =  75000.0

Select Operation done successfully
```

كيف تعمل...

وفيما يلي الجزء الفرعي من التعليمات البرمجية التي تتناول أساسا اختيار السجلات من جدول:

```
cur.execute('SELECT id, name, address, salary  from EMPLOYEES')
rows = cur.fetchall()
for row in rows:
    print 'ID = ', row[0]
    print 'NAME = ', row[1]
    print 'ADDRESS = ', row[2]
    print 'SALARY = ', row[3], '\n'

print 'Select Operation done successfully'
conn.close()
```

في التعليمات البرمجية السابقة، أولا قمنا باستدعاء دالة كائن المؤشر ()execute لتنفيذ جملة SELECT. ومع ذلك، فالوضع هنا هو أن جدولا معينا قد يتكون من سجلات متعددة وهدفنا هو جلب تلك الصفوف المتعددة من خلال دمج كل صف، وهذا سجل واحد في وقت واحد. لتحقيق هدف جلب صفوف متعددة من جدول نستخدم الدالة ()fetchall في كائن المؤشر الذي يقوم بإرجاع كافة الصفوف من resultset، وبالتالي إرجاع قائمة الصفوف من جدول. لنسخ كل صف نستخدم حلقة for للمرور على كل الصفوف في الجدول وطباعة كل صف من الجدول على سطر الأوامر في كل لفة.

يمكنك الرجوع إلى رابط الويب التالي للحصول على مزيد من المعلومات:

https://wiki.postgresql.org/wiki/Psycopg2_Tutorial

تحديث السجلات باستخدام بايثون

هنا، في هذه الوصفة نحن نريد تحديث السجلات الموجود ة في جدول في قاعدة بيانات بوستجريسكل باستخدام لغة بايثون.

الاستعداد للعمل

في هذه الوصفة، سنقوم أولا بعرض عدد السجلات الموجودة في الجدول، ثم سنقوم بتحديث بعض السجلات، ورؤية عدد السجلات التي حدثت ومن ثم نرى كيف أصبحت السجلات المحدثة مرئية في الجدول عندما يوصل إلى الجدول مرة أخرى.

كيف ينجز ذلك...

أولا نتحقق من السجلات الموجودة في الجدول EMPLOYEES.

```
dvdrental=# select * from employees;
 id |      name       | age | address                          | salary
----+----------------+-----+----------------------------------+--------
  1 | SandeepSingh   |  39 | Saharanpur                       |  90000
  2 | NeerajKumar    |  42 | Rohtak                           |  90000
  3 | AmitGovil      |  37 | Aligarh                          |  88000
  4 | SandeepSharma  |  36 | Haridwar                         |  75000
```

بعد ذلك نستخدم شفرة بايثون التالية لتحديث بعض السجلات الموجودة في الجدول EMPLOYEES وحفظ التعليمات البرمجية التالية في ملف يسمى update.py:

```
#!/usr/bin/python

import psycopg2

conn = psycopg2.connect(database='dvdrental', user='postgres',
                        password='postgres', host='127.0.0.1',
                        port='5432')
print 'Opened database successfully'

cur = conn.cursor()

cur.execute('UPDATE EMPLOYEES SET  SALARY = 105000.00 WHERE ID=1')
conn.commit()
print 'Total number of rows updated :', cur.rowcount

cur.execute('SELECT id, name, address, salary  FROM  EMPLOYEES')
rows = cur.fetchall()
for row in rows:
    print 'ID = ', row[0]
    print 'NAME = ', row[1]
    print 'ADDRESS = ', row[2]
    print 'SALARY = ', row[3], '\n'

print 'Update Operation done successfully'
conn.close()
```

في التعليمات البرمجية السابقة، نحن نستخدم عبارة UPDATE لتعيين الراتب إلى ٥٥٠٠٠ حيث قيمة العمود ID هي ١. ثم نعرض السجلات المحدثة والتأكد من أن السجلات المحدثة أصبحت مرئية في الجدول EMPLOYEES، كما هو مبين في الخرج التعليمات البرمجية التالية:

```
--bash-3.2$ python update.py
Opened database successfully
Total number of rows updated : 1
ID =  2
NAME =  NeerajKumar
ADDRESS =  Rohtak
SALARY =  90000.0

ID =  3
NAME =  AmitGovil
ADDRESS =  Aligarh
SALARY =  88000.0
```

```
ID =  4
NAME =  SandeepSharma
ADDRESS =  Haridwar
SALARY =  75000.0

ID =  1
NAME =  SandeepSingh
ADDRESS =  Saharanpur
SALARY =  105000.0

Update Operation done successfully
```

كيف تعمل...

فيما يلي مقتطف من التعليمات البرمجية التي استخدمت لتحديث السجلات الموجودة باستخدام لغة بايثون في جدول الموظفين:

```
cur = conn.cursor()

cur.execute('UPDATE EMPLOYEES SET  SALARY = 105000.00 WHERE ID=1')
conn.commit()
print 'Total number of rows updated :', cur.rowcount
```

إذا ألقينا نظرة على التعليمات البرمجية السابقة، فإنا سنكون على دراية بوظيفة مؤشر كائن الاتصال cursor() مع وظيفة كائن المؤشر execute() وقد نوقشت هذه في الوصفات السابقة. ومع ذلك، لتحديث السجلات في جدول، فإن التغيير الأول الذي يحدث هنا هو استخدام عبارة UPDATE كجزء من الدالة () execute لتحديث سجل أساسي في الجدول. ومع ذلك، نحن بحاجة للتأكد من أن التغييرات التي أجريناها في الجدول مرئية عبر قاعدة البيانات. من الآن فصاعدا، نستخدم الدالة commit() لحفظ التغييرات التي أجريت بواسطة عبارة UPDATE. وبمجرد القيام بذلك، فإن التغييرات التي أجريت بواسطة عبارة UPDATE صارت مرئية لأي شخص يختار البيانات من الجدول. كما نستخدم سمة "عدد الصفوف rowcount" لعنصر المؤشر لمعرفة عدد السجلات التي عدلت بواسطة آخر عبارة نفذت. يمكن استخدام السمة rowcount إما مع عبارات UPDATE أو DELETE أو INSERT.

حذف السجلات باستخدام بايثون

هنا في هذه الوصفة سنقوم بعرض كيفية حذف السجلات في جدول باستخدام لغة بايثون.

الاستعداد للعمل

في هذه الوصفة، سنقوم أولا بعرض عدد السجلات الموجودة في الجدول، ثم سنقوم بحذف بعض السجلات، ومراجعة عدد السجلات المحذوفة ثم نرى العدد المتاح من السجلات الموجودة في الجدول بعد الحذف.

كيف ينجز ذلك...

فيما يلي خطوات حذف السجلات في جدول:

١. أولا سوف نتحقق من السجلات الموجودة في الجدول EMPLOYEES.

```
dvdrental=# select * from employees;
 id |      name      | age | address              | salary
----+----------------+-----+----------------------+--------
  1 | SandeepSingh   |  39 | Saharanpur           |  90000
  2 | NeerajKumar    |  42 | Rohtak               |  90000
  3 | AmitGovil      |  37 | Aligarh              |  88000
  4 | SandeepSharma  |  36 | Haridwar             |  75000
```

٢. بعد ذلك سنستخدم شفرة بايثون التالية لحذف بعض السجلات من جدول EMPLOYEES . نحفظ الشفرة في ملف يسمى delete.py، والذي سنقوم بتنفيذه من سطر الأوامر:

```
#!/usr/bin/python

import psycopg2

conn = psycopg2.connect(database='dvdrental', user='postgres',
                        password='postgres', host='127.0.0.1',
                        port='5432')
print 'Opened database successfully'

cur = conn.cursor()

cur.execute('DELETE from EMPLOYEES where ID=2;')
```

```
conn.commit()
print 'Total number of rows deleted :', cur.rowcount

cur.execute('SELECT id, name, address, salary  from EMPLOYEES')
rows = cur.fetchall()
for row in rows:
    print 'ID = ', row[0]
    print 'NAME = ', row[1]
    print 'ADDRESS = ', row[2]
    print 'SALARY = ', row[3], '\n'

print 'DELETE Operation done successfully'
conn.close()
```

هنا، إن عبارة DELETE التي استخدمناها، تستخدم لحذف سجل من الجدول حيث قيمة العمود ID هو ٢.

٣. ثم نرى خرج سطر الأوامر من التعليمات البرمجية السابقة:

```
bash-3.2$ python delete.py
Opened database successfully
Total number of rows deleted : 1
ID =  1
NAME =  SandeepSingh
ADDRESS =  Saharanpur
SALARY =  90000.0

ID =  3
NAME =  AmitGovil
ADDRESS =  Aligarh
SALARY =  88000.0

ID =  4
NAME =  SandeepSharma
ADDRESS =  Haridwar
SALARY =  75000.0

DELETE Operation done successfully
```

كيف تعمل...

مقتطف التالي من التعليمات البرمجية التي استخدمت لحذف السجلات من الجدول EMPLOYEES:

```
cur = conn.cursor()

cur.execute('DELETE from EMPLOYEES where ID=2;')
conn.commit()
print 'Total number of rows deleted :', cur.rowcount
```

في مقتطف التعليمات البرمجية السابق، نستخدم الدالة execute كائن المؤشر لتنفيذ عبارة DELETE. ثم نستخدم دالة commit كائن الاتصال لعمل أو حفظ التغييرات التي أجريت بواسطة عبارة DELETE. وأخيرا، نستخدم السمة rowcount كائن المؤشر للعثور على عدد السجلات التي حذفت بواسطة آخر عبارة DELETE المنفذة.

١٢ - ترحيل البيانات من قواعد البيانات الأخرى وتحديث مجموعة بوستجريسكل

في هذا الفصل، سنغطي الوصفات التالية:

- استخدام pg_dump لترقية البيانات
- استخدام الأداة pg_upgrade لترقية الإصدار
- نسخ البيانات من قواعد البيانات الأخرى إلى بوستجريسكل باستخدام GoldenGate

مقدمة

في كثير من الأحيان في مهنة مسؤول قاعدة البيانات، يطلب القيام بترقيات للنسخة الرئيسية من خادم بوستجريسكل. وعلى مدى فترة من الزمن تضاف مصطلحات جديدة ومميزات إلى بوستجريسكل وهذا يؤدي إلى إصدار نسخة رئيسية جديدة. وللحصول على المميزات الجديدة في الإصدار الجديد، نحتاج إلى ترقية بوستجريسكل الحالية إلى الإصدارة الجديدة. تتطلب ترقيات قاعدة البيانات التخطيط السليم والتنفيذ الدقيق والتوقف المخطط له. تقدم بوستجريسكل طريقتين رئيسيتين للقيام بترقية الإصدار.

- بمساعدة من أداة pg_dump
- بمساعدة النص البرمجي pg_upgrade

في هذا الفصل أيضا سنغطي أداة أوراكل غولدنغايت Oracle GoldenGate. وغولدنغايت GoldenGate هو برنامج النسخ المتماثل غير المتجانس الذي يمكن استخدامه لنسخ البيانات بين قواعد البيانات المختلفة.

في هذا الفصل سوف نقوم بتغطية النسخ المتماثلة غير المتجانسة بين أوراكل و بوستجريسكل.

استخدام pg_dump لترقية البيانات

هنا، في هذه الوصفة، سنقوم بترقية بوستجريسكل من الإصدار ٩.٢ إلى ٩.٣ وسوف نستخدم أداة pg_dump لهذا الغرض.

الاستعداد للعمل

المتطلبات الأساسية هنا هي أنه يجب أن يكون عنقود بوستجريسكل الحالي معدا ومشغلا.

النسخة المطلوبة هنا هي بوستجريسكل الإصدار ٩.٢. نفذت هذه الخطوات على جهاز سنتوس ٦٤ بايت.

كيف ينجز ذلك...

فيما يلي سلسلة من الخطوات التي تحتاج إلى القيام بها لترقية بوستجريسكل من الإصدار ٩.٢ إلى ٩.٣ باستخدام pg_dump:

١. قم بالنسخ الاحتياطي لقاعدة البيانات الخاصة بك باستخدام الأمر pg_dumpall.

```
pg_dumpall >  db.backup
```

٢. الخطوة التالية ستكون إيقاف خادم بوستجريسكل الحالي.

```
pg_ctl -D /var/lib/pgsql/9.2/data  stop
```

٣. الخطوة التالية هي إعادة تسمية دليل التثبيت بوستجريسكل القديم.

```
mv /var/lib/pgsql     /var/lib/pgsql.old
```

٤. الخطوة التالية ستكون لتثبيت النسخة الجديدة من بوستجريسكل وهو بوستجريسكل الإصدار ٩.٣. قبل القيام بذلك، سوف نتحقق من الحزم الموجودة قبل تثبيت تلك الجديدة مع الأمر التالي ومن ثم سوف نقوم بتثبيت حزمة الإصدار الجديد:

```
rpm -qa |grep postgresql
wget http://yum.postgresql.org/9.3/redhat/rhel-6.4-x86_64/pgdg-
centos93-9.3-1.noarch.rpm

rpm -Uvh ./pgdg- centos93-9.3-1.noarch.rpm
```

```
yum install postgresql93-server.x86_64 postgresql93-contrib.x86_64
postgresql93-libs.x86_64 postgresql93.x86_64 postgresql93-devel.
x86_64
```

٥. الخطوة التالية ستكون إعداد خادم الإصدار ٩.٣ بوستجريسكل.

```
/usr/pgsql-9.3/bin/initdb -D    /var/lib/pgsql/9.3/data
```

٦. بمجرد أن يتم ضبط عنقود قاعدة البيانات، فإن الخطوة التالية هي استعادة ملفات الضبط من مجلد بيانات الإصدار ٩.٢ السابق إلى الإصدار الحالي من مجلد موقع البيانات ٩.٣.

```
cd /var/lib/pgsql.old/9.2/data
cp  pg_hba.conf postgresql.conf  /var/lib/pgsql/9.3/data
```

٧. الخطوة التالية ستكون لبدء خادم قاعدة البيانات بوستجريسكل ٩.٣.

```
pg_ctl -D /var/lib/pgsql/9.3/data  start
```

٨. وأخيرا، كخطوة أخيرة، استعادة البيانات الخاصة بك من النسخة الاحتياطية التي أنشئت في الخطوة ١.

```
/usr/pgsql-9.3/bin/psql    -d postgres     -f  db.backup
```

٩. كخطوة تالية، يمكننا إما إزالة دليل بيانات الإصدار القديم أو يمكننا مواصلة العمل جنبا إلى جنب مع إصدارات الخادم.

١٠. إذا اخترنا إزالة الإصدار القديم، كما هو مذكور في الخطوة ٩، يمكننا بعد ذلك إزالة حزم الإصدار القديم منها على النحو التالي:

```
yum remove postgresql92-server-9.2.3-2PGDG.rhel6.x86_64 postgresql92-contrib-
9.2.3-2PGDG.rhel6.x86_64 postgresql92-libs-9.2.3-2PGDG.rhel6.x86_64
postgresql92-9.2.3-2PGDG.rhel6.x86_64 postgresql92-devel-9.2.3-
2PGDG.rhel6.x86_64
```

كيف تعمل...

في البداية سنأخذ تفريغ جميع قواعد البيانات في القائمة الحالية الإصدار ٩.٢ من بوستجريسكل. ثم نبدأ بإيقاف نظيف لخادم بوستجريسكل الحالي وإعادة تسمية مجلد التثبيت بوستجريسكل الحالية لتجنب أي تعارض مع الإصدار الجديد من بوستجريسكل، وهو الإصدار ٩.٣ الذي سنثبته. عندما تثبت حزم الإصدار الجديد، نبدأ بضبط مجلد قاعدة بيانات لخادم بوستجريسكل ٩.٣. للتأكد من أن إعدادات التضبيط المطلوبة نافذة المفعول، سوف نحتاج إلى نسخ ملفات التضبيط من مجلد بيانات الإصدار القديم إلى مجلد بيانات الإصدار الجديد ثم نقوم بتشغيل الإصدار الجديد لخدمة خادم بوستجريسكل باستخدام إعدادات التضبيط التي عرفت لبيئة الخادم القديم. عندما يبدأ خادم بوستجريسكل الإصدار ٩.٣ يمكننا الاتصال بقواعد البيانات على هذا الخادم. في نهاية المطاف نقوم باستعادة كافة الجداول وقواعد البيانات من خادم بوستجريسكل القديم ٩.٢ إلى خادم بوستجريسكل الإصدار ٩.٣ الجديد باستخدام النسخ الاحتياطي الذي قمنا به في الخطوة ١ في القسم السابق.

يمكنك الرجوع إلى الرابط التالي للحصول على شرح أكثر تفصيلا حول ترقية مجموعة بوستجريسكل باستخدام pg_dump:

http://www.postgresql.org/docs/9.3/static/upgrading.html

استخدام الأداة pg_upgrade لترقية الإصدار

في هذه الوصفة، سوف نتحدث عن ترقية عنقود بوستجريسكل باستخدام أداة pg_upgrade. وسنقوم بتغطية نسخة بوستجريسكل من الإصدار ٩.٢ إلى الإصدار ٩.٣.

الاستعداد للعمل

المتطلبات الأساسية الوحيدة هنا هي أن عنقود بوستجريسكل يجب إعداده وتشغيله. النسخة المطلوبة هنا هي بوستجريسكل الإصدار ٩.٢. نفذت هذه الخطوات على جهاز سينتوس ٦٤ بت.

كيف ينجز ذلك...

فيما يلي خطوات ترقية عنقود بوستجريسكل من الإصدار ٩.٢ إلى الإصدار ٩.٣ باستخدام الأداة المساعدة pg_upgrade:

١. اصنع نسخة احتياطية كاملة من مجلد البيانات باستخدام تفريغ نظام الملفات أو استخدام pg_dumpall لأخذ نسخة احتياطية للبيانات. وقبل أخذ نسخة احتياطية يجب إيقاف تشغيل خادم بوستجريسكل.

```
pg_ctl     -D $PGDATA stop

cd /var/lib/pgsql/9.2/

tar -cvf data.tar data
```

٢. الخطوة التالية ستكون تثبيت الإصدار الجديد من بوستجريسكل.

```
wget http://yum.postgresql.org/9.3/redhat/rhel-6.4-x86_64/pgdg-
centos93-9.3-1.noarch.rpm
rpm -ivh ./pgdg- centos93-9.3-1.noarch.rpm
```

٣ - وبما أن المستودع مثبت الآن، فإن الخطوة التالية هي تحديد الحزم التي يلزم تركيبها. لهذا الغرض، تحقق من الحزم التي استخدمت للإصدار الحالي ومن ثم الحصول على قائمة الحزم التي تحتاج إلى تثبيت الإصدار الجديد بوستجريسكل ٩.٣.

```
 rpm -qa | grep postgre | grep 92
postgresql92-server-9.2.3-2PGDG.rhel6.x86_64
postgresql92-contrib-9.2.3-2PGDG.rhel6.x86_64
postgresql92-libs-9.2.3-2PGDG.rhel6.x86_64
postgresql92-9.2.3-2PGDG.rhel6.x86_64
postgresql92-devel-9.2.3-2PGDG.rhel6.x86_64


 yum list postgres* | grep 93
postgresql93.x86_64 9.3.4-1PGDG.rhel6 pgdg93
postgresql93-contrib.x86_64 9.3.4-1PGDG.rhel6 pgdg93
postgresql93-debuginfo.x86_64 9.3.4-1PGDG.rhel6 pgdg93
postgresql93-devel.x86_64 9.3.4-1PGDG.rhel6 pgdg93
postgresql93-docs.x86_64 9.3.4-1PGDG.rhel6 pgdg93
postgresql93-jdbc.x86_64 9.3.1100-1PGDG.rhel6 pgdg93
postgresql93-jdbc-debuginfo.x86_64 9.3.1100-1PGDG.rhel6 pgdg93
postgresql93-libs.x86_64 9.3.4-1PGDG.rhel6 pgdg93
postgresql93-odbc.x86_64 09.02.0100-1PGDG.rhel6 pgdg93
postgresql93-odbc-debuginfo.x86_64 09.02.0100-1PGDG.rhel6 pgdg93
postgresql93-plperl.x86_64 9.3.4-1PGDG.rhel6 pgdg93
postgresql93-plpython.x86_64 9.3.4-1PGDG.rhel6 pgdg93
postgresql93-pltcl.x86_64 9.3.4-1PGDG.rhel6 pgdg93
postgresql93-server.x86_64 9.3.4-1PGDG.rhel6 pgdg93
postgresql93-test.x86_64 9.3.4-1PGDG.rhel6 pgdg93
```

الحزم التي ستثبت للإصدار الجديد سوف تتطابق مع الحزم المثبتة حاليا للإصدار القديم.

```
yum install postgresql93-server.x86_64 postgresql93-contrib.x86_64
postgresql93-libs.x86_64 postgresql93.x86_64 postgresql93-devel.x86_64
```

٤. الآن بعد تثبيت الإصدار الجديد من بوستجريسكل، فإن الخطوة التالية هي تضبيط مجلد البيانات لقاعدة بيانات بوستجريسكل ٩.٣ الجديدة.

```
/etc/init.d/postgresql-9.3 initdb
```

٤. بمجرد أن يتم تضبيط مجلد البيانات لبوستجريسكل الإصدار ٩.٣ الجديد، فإن الخطوة التالية هي تشغيل الأداة المساعدة pg_upgrade.

```
cd    /usr/pgsql-9.3/bin

./pg_upgrade -v -b /usr/pgsql-9.2/bin/ -B /usr/pgsql-9.3/bin/ -d /
var/lib/pgsql/9.2/data/ -D /var/lib/pgsql/9.3/data/
```

بمجرد اكتمال الترقية، فإنه سيقوم بإنشاء ملفين analy_new_cluster.sh وملف delete_old_cluster.sh على التوالي.ستستخدم هذه الملفات أساسا لإنشاء إحصاءات محسنة وحذف ملفات بيانات إصدار عنقود بوستجريسكل القديم.

٦. بعد خطوة الترقية سنحتاج إلى نسخ ملفات التضبيط وملفات التوثيق الموجودة في الإصدار القديم إلى الإعداد الجديد، كما يلي:

```
cd /var/lib/pgsql/9.2/data
cp -p pg_hba.conf postgresql.conf /var/lib/pgsql/9.3/data/
```

٧. الخطوة التالية ستكون تشغيل خدمة خادم بوستجريسكل، الإصدار ٩.٣

```
service postgresql-9.3 start
```

٨. الخطوة التالية هي تشغيل البرنامج النصي analyze_new_cluster.sh الذي أنشئ في نهاية الخطوة ٥. ويستخدم هذا البرنامج النصي لجمع إحصاءات الحد الأدنى المحسنة من أجل الحصول على أداء ونظام بوستجريسكل صالح للاستخدام.

```
./analyze_new_cluster.sh
```

٩. الخطوة التالية هي إزالة دليل بوستجريسكل القديم عن طريق تشغيل البرنامج النصي التالي:

```
./delete_old_cluster.sh
```

١٠. وأخيرا، كخطوة أخيرة، سوف نقوم بإزالة حزم بوستجريسكل الإصدار ٩.٢ تثبيت القديمة.

```
yum remove postgresql92
```

كيف تعمل...

بعد تثبيت الحزم لخادم بوستجريسكل الإصدار ٩.٣، ما نقوم به هنا هو تغيير موقع مجلد البيانات في البرنامج النصي لبدء التشغيل، كما هو مبين في الخطوة ٤. بعد هذا نقوم بتضبيط دليل البيانات لخادم بوستجريسكل الجديد. الفرق في الخطوات هنا وبين الوصفة السابقة، أننا نقوم بتغيير موقع مجلد البيانات ومسار السجل، ورقم المنفذ لخادم بوستجريسكل الجديد، بينما في وصفة سابقة أعيدت تسمية إصدارة مجلد البيانات لخادم بوستجريسكل للإصدارة الجديدة . عندما يكتمل إعداد دليل البيانات، نوقف خادم بوستجريسكل الحالي ثم نقوم بتشغيل البرنامج النصي pg_upgrade لترقية الإعداد الحالي إلى الإصدار الجديد. يتطلب النص البرمجي pg_upgrade تحديد مسار مجلدات البيانات القديمة والجديدة. بمجرد اكتمال الترقية فإنه يولد اثنين من البرامج النصية analy_new_cluster.sh و delete_old_cluster.sh لإنشاء إحصاءات وحذف النسخة القديمة لدليل بوستجريسكل. للحفاظ على التضبيط الحالي، سنحتاج إلى ملفات pg_hba.conf و postgresql.conf من مجلد بيانات الإصدار القديم إلى مجلد البيانات الإصدار الجديد، كما هو موضح في الخطوة ٦ في القسم السابق ومن ثم يمكننا بدء تشغيل خادم ترقية بوستجريسكل.

بمجرد بدء تشغيل الخادم، يمكننا بعد ذلك المضي قدما لإنشاء إحصاءات عبر البرنامج النصي analy_new_cluster.sh ثم نقوم بإزالة دليل الإصدار القديم عن طريق البرنامج النصي delete_old_cluster.sh، كما هو مبين في الخطوتين ٨ و ٩ على التوالي.

يمكنك الرجوع إلى رابط الويب التالي للحصول على مزيد من المعلومات حول عملية الترقية:

http://www.postgresql.org/docs/9.3/static/pgupgrade.html
http://no0p.github.io/postgresql/2014/03/29/upgrading-pgubuntu.html

نسخ البيانات من قواعد بيانات أخرى إلى بوستجريسكل باستخدام GoldenGate

في هذه الوصفة، سنقوم بتغطية النسخ المتماثل غير المتجانس باستخدام برنامج أوراكل غولدنغايت. وسنقوم بترحيل بيانات الجدول من أوراكل إلى بوستجريسكل.

الاستعداد للعمل

وبما أن هذه الوصفة تتحدث عن نسخ البيانات من أوراكل إلى بوستجريسكل، فمن المهم تغطية تثبيت أوراكل. أيضا، وبما أن غولدنغايت هي الأداة الأساسية المستخدمة، سنغطي أيضا تركيب غولدنغايت لكل من أوراكل و بوستجريسكل.

لتثبيت برنامج أوراكل 11g على منصة لينكس، يمكنك الرجوع إلى أي من روابط الويب التالية:

http://oracle-base.com/articles/11g/oracle-db-11gr2-installation-on-oracle-linux-5.php
http://dbaora.com/install-oracle-11g-release-2-11-2-on-centos-linux-7/

لتثبيت غولدنغايت لقاعدة بيانات أوراكل، راجع رابط الويب التالي:

http://docs.oracle.com/cd/E35209_01/doc.1121/e35957.pdf

وهنا سنستعرض خطوات التثبيت العامة للتسهيل على القارئ. يرجى الرجوع إلى رابط الويب السابق للحصول على مزيد من المعلومات التفصيلية:

- قم بالتسجيل الدخول إلى edelivery.oracle.com.

- حدد Oracle Fusion Middleware من القائمة المنسدلة Select a Product Pack وحدد الخيار Linux x86-64 من القائمة المنسدلة ب Platform وانقر على زر Go.

- اختر الخيار Oracle GoldenGate على Oracle v11.2.1 Media Pack لـ Linux x86-64، انقر على زر متابعة، وسوف تفتح وصلة على شبكة الإنترنت. ثم قم بتحميل الملف باسم Oracle GoldenGate V11.2.1.0.3 for Oracle 11g on Linux x86-64..

- كخطوة تالية، قم باستخراج الملف الذي نزلته وقم بتغيير الدليل إلى الموقع الجديد ثم قم بتشغيل واجهة سطر الأوامر GoldenGate باستخدام الأمر ggsci. قبل إطلاق واجهة سطر الأوامر غولدنغايت قم بتنصيب مجلد التثبيت غولدنغايت ومسار المكتبة في متغيرات PATH و LD_LIBRARY_PATH على التوالي.

لتثبيت غولدنغايت لبوستجريسكل، راجع رابط الويب التالي:

https://docs.oracle.com/cd/E35209_01/doc.1121/e29642.pdf

وهنا سنستعرض خطوات التثبيت العامة للتسهيل على القارئ. يرجى الرجوع إلى رابط الويب السابق للحصول على مزيد من المعلومات التفصيلية:

- قم بالتسجيل الدخول إلى edelivery.oracle.com.

- حدد Oracle Fusion Middleware من القائمة المنسدلة Select a Product Pack وحدد الخيار Linux x86-64 من القائمة المنسدلة ب Platform وانقر على زر Go.

- اختر الخيار **Oracle GoldenGate for Non Oracle Database v11.2.1 Media Pack for Linux x86-64**، انقر على زر متابعة، وسوف تفتح وصلة على شبكة الإنترنت. ثم قم بتحميل الملف باسم **Oracle GoldenGate V11.2.1.0.2 for PostgreSQL on Linux x86-64**.

- كخطوة تالية، قم باستخراج الملف الذي نزلته وقم بتغيير الدليل إلى الموقع الجديد ثم قم بتشغيل واجهة سطر الأوامر GoldenGate باستخدام الأمر ggsci. قبل إطلاق واجهة سطر الأوامر غولدنغايت قم بتنصيب مجلد التثبيت غولدنغايت ومسار المكتبة في متغيرات PATH و LD_LIBRARY_PATH على التوالي.

في هذا القسم، سنقوم أولا بتغطية نظرة عامة موجزة عن الإجراء المستخدم لنسخ بيانات الجدول من قاعدة بيانات المصدر، وهي هنا أوراكل، إلى قاعدة البيانات الهدف بوستجريسكل.

١. أولا ، سيكون علينا إنشاء أدلة فرعية مختلفة لغولدنغايت وملفات تعريف قاعدة البيانات المختلفة من قاعدة بيانات المصدر أوراكل.
٢. الخطوة التالية هي إنشاء ملف إعدادات يحتوي على رقم منفذ لعملية مدير غولدنغايت على قاعدة بيانات المصدر ثم قم بتشغيل المدير.
٣. على غرار هذا على قاعدة البيانات المستهدفة، وهي هنا بوستجريسكل، سيكون علينا إنشاء أدلة فرعية مختلفة لغولدنغايت وملفات تعريف قاعدة البيانات المختلفة.
٤. الخطوة التالية هي إنشاء ملف إعدادات يحتوي على رقم منفذ لعملية مدير غولدنغايت على قاعدة البيانات المستهدفة ثم قم بتشغيل المدير.
٥. الخطوة التالية هي إنشاء جدولين بنفس البنية على كل من المصدر، وهي أوراكل، وقاعدة بيانات الهدف، وهي بوستجريسكل.
٦. الآن لدينا الجداول التي أنشئت على كل من قاعدة البيانات المصدر والهدف، وسوف نقوم بتسجيل الدخول إلى قاعدة بيانات المصدر من أداة غولدنغايت والتقاط تعريفات الجدول للجداول التي تحتاج إلى نسخ.
٧. على غرار الخطوة السابقة سوف نقوم بتسجيل الدخول إلى قاعدة البيانات المستهدفة باستخدام واجهة سطر الأوامر غولدنغايت والتقاط تعريفات الجدول للجدول الذي أنشيء في الخطوة ٥.
٨. في الخطوة التالية، نبدأ عملية الاستخراج على المصدر. نقوم أولا بإنشاء ملف إعدادات لعملية الاستخراج الذي يحتوي على معلومات حول المضيف البعيد ويتكون من ملف التعقب الذي يستخدم لالتقاط أي تغييرات على

الجدول في قاعدة بيانات المصدر ونقل هذه التغييرات إلى قاعدة البيانات المستهدفة. ثم نبدأ عملية الاستخراج وسوف نقوم بالتقاط أي تغييرات على الجدول في قاعدة بيانات المصدر، وهي أوراكل.

٩. والخطوة التالية هي بدء عملية النسخ المتماثل على قاعدة البيانات المستهدفة. لهذا قمنا بإعداد ملف إعدادات النسخ المتماثل replicat . بمجرد بدء عملية النسخ المتماثل، ستقرا التغييرات من ملف التعقب الذي استخدم في الخطوة السابقة في المصدر لالتقاط التغييرات التي أجريت على الجدول في قاعدة بيانات المصدر. ستقرأ عملية النسخ المتماثل replicat هذه التغييرات وتقوم بتفريغها في قاعدة البيانات المستهدفة، وهذا هي بوستجريسكل.

١٠. الآن وبعد إعداد عملية الاستخراج على قاعدة بيانات المصدر لالتقاط التغييرات و إعداد عملية النسخ المتماثل على قاعدة البيانات الهدف لقراءة تلك التغييرات. سنبدأ الآن بإضافة / تغيير بعض السجلات على المصدر. ومع التقاط عملية استخراج هذه التغييرات وتسجلها في ملف التعقب ثم تشحن إلى خادم المستضيف لقاعدة البيانات الهدف، تقرأ عملية النسخ المتماثل المقيمة في الهدف تلك التغييرات من ملف التعقب وتطبقها على قاعدة البيانات المستهدفة.

نحن نفترض أن اسم مستخدم nkumar مع كلمة المرور nkumar قد أعدتا على حد سواء في أوراكل وبوستجريسكل. ولذلك سنقوم باستخدام الجداول التي أنشأت في مخطط nkumar للنسخ المتماثل بين أوراكل وبوستجريسكل.

على سبيل المثال، على أوراكل، يمكننا إنشاء مستخدم المخطط nkumar، على النحو التالي، بعد تسجيل الدخول كمستخدم sys:

```
SQL> CREATE USER nkumar identified by nkumar;
SQL> GRANT CREATE ANY TABLE to nkumar;
```

لإنشاء مستخدم nkumar في بوستجريسكل، يمكنك الرجوع إلى الفصل ١، إدارة قواعد البيانات وخادم بوستجريسكل لمزيد من التفاصيل حول كيفية إنشاء مستخدم في بوستجريسكل وبناء على ذلك تقوم بإنشاء هذا المستخدم في بوستجريسكل.

كيف ينجز ذلك...

فيما يلي التسلسل الكامل للخطوات المطلوبة لترحيل بيانات الجدول / التغييرات من أوراكل إلى بوستجريسكل باستخدام غولدنغايت:

١. أولا قم بالاتصال كمستخدم SYS باستخدام مصادقة نظام التشغيل على الجهاز استضافة قاعدة بيانات أوراكل باستخدام الأداة المساعدة sqlplus. تستخدم الأداة مصادقة نظام التشغيل بشكل افتراضي، لذا لا يلزم تحديد كلمة المرور. في وقت تثبيت أوراكل، فإنه عادة ما يطلب من المستخدم تغيير كلمة المرور، ولكن إذا لم تحدد كلمة المرور يمكنك استخدام كلمة المرور الافتراضية change_on_install.

```
sqlplus / as sysdba
```

٢. بمجرد تسجيل الدخول إلى قاعدة بيانات أوراكل قم بإجراء التغييرات التالية:

```
SQL> alter system set log_archive_dest_1='LOCATION=/home/abcd/oracle/oradata/arch';
```

٣. لجعل التغييرات المذكورة أعلاه في حيز التنفيذ، قم بإيقاف تشغيل وإعادة تشغيل قاعدة بيانات أوراكل.

```
SQL> shutdown immediate
SQL> startup mount
```

٤. قم بضبط الأرشفة على قاعدة بيانات أوراكل للتأكد من أن التغييرات التي تجريها المعاملات تلتقط وتسجل في ملفات archivelog.

```
SQL> alter database archivelog;
SQL> alter database open;
```

٥ - وتتمثل الخطوة التالية في تفعيل الحد الأدنى من السجل التكميلي.

```
SQL> alter database add supplemental log data;
SQL> alter database force logging;
SQL> SELECT force_logging, supplemental_log_data_min FROM v$database;
FOR SUPPLEME
--- --------
YES      YES
```

٦- الخطوة التالية هي إضافة مسار الدليل غولدنغايت إلى PATH ومسار المكتبة إلى متغيرات البيئة LD_LIBRARY_PATH على التوالي.

```
export PATH=$ORACLE_HOME/bin:$ORACLE_HOME/OPatch:$HOME/ggs:$PATH
export LD_LIBRARY_PATH=$ORACLE_HOME/lib:$HOME/ggs/lib
```

٧ - وتتمثل الخطوة التالية في إطلاق واجهة سطر الأوامر غولدنغايت لـ أوراكل.

```
./ggsci
```

٨ - والخطوة التالية هي إنشاء أدلة فرعية مختلفة ل غولدنغايت مثل مجلدات لملفات التقرير، وتعريف قاعدة البيانات وما إلى ذلك.

```
GGSCI> create subdirs

Creating subdirectories under current directory /home/abcd/oracle/ggs

Parameter files               /home/abcd/oracle/ggs/dirprm: already exists
Report files                  /home/abcd/oracle/ggs/dirrpt: created
Checkpoint files              /home/abcd/oracle/ggs/dirchk: created
Process status files          /home/abcd/oracle/ggs/dirpcs: created
SQL script files              /home/abcd/oracle/ggs/dirsql: created
Database definitions files    /home/abcd/oracle/ggs/dirdef: created
Extract data files            /home/abcd/oracle/ggs/dirdat: created
Temporary files               /home/abcd/oracle/ggs/dirtmp: created
Stdout files                  /home/abcd/oracle/ggs/dirout: created
```

٩. الخطوة التالية هي إنشاء ملف إعدادات للمدير يحتوي على رقم منفذ للمدير. هنا، ندخل المنفذ ٧٨٠٩ كعدد المنفذ.

```
GGSCI > edit param mgr
GGSCI > view param mgr
PORT 7809
```

١٠. الخطوة التالية ستكون الخروج من المدير، وتشغيله المدير ومن ثم التحقق مما إذا كان قيد التشغيل.

```
 GGSCI > startw mgr
GGSCI > info all

Program     Status      Group       Lag at Chkpt  Time Since Chkpt
MANAGER     RUNNING

GGSCI > info mgr
Manager is running (IP port 7809).
```

١١. الخطوة التالية ستكون تسجيل الدخول إلى الخادم الذي يستضيف خادم بوستجريسكل وجعل خطوات التضبيط غولدنغايت هناك. قم أولا بإضافة الدليل غولدنغايت إلى متغيرات البيئة LD_LIBRARY_PATH و باث.

```
export LD_LIBRARY_PATH=/usr/pgsql/lib:/usr/pgsql/ggs/lib
export PATH=/usr/pgsql/bin:/usr/pgsql/ggs:$PATH
```

١٢. يستخدم غولدنغايت اتصال ODBC للاتصال بقاعدة بيانات postgres. الخطوة التالية هي إنشاء ملف ODBC.

يشحن برنامج تشغيل ODBC جنبا إلى جنب مع التثبيت على لينكس / يونكس، يجب عليك فقط إنشاء ملف الإعداد.

في حالة عدم توفر برنامج تشغيل ODBC، يمكنك الرجوع إلى رابط الويب التالي لتحميل برنامج التشغيل بوستجريسكل:

http://www.uptimemadeeasy.com/linux/install-postgresql-odbc-driver-on-linux/

```
 view odbc.ini

[ODBC Data Sources]
GG_Postgres=DataDirect 6.1 PostgreSQL Wire Protocol
[ODBC]
IANAAppCodePage=106
InstallDir=/usr/pgsql/ggs
[GG_Postgres]
Driver=/usr/pgsql/ggs/lib/GGpsql25.so
Description=DataDirect 6.1 PostgreSQL Wire Protocol
Database=test
HostName=dbtest
PortNumber=5432
LogonID=nkumar
Password=nkumar
```

١٣. الخطوة التالية ستكون تصدير متغير بيئة ODBC ، حيث متغير ODBCINI يجب أن يشير إلى الملف odbc.ini الذي أنشأناه في الخطوة السابقة.

يمكن كذلك تعيين هذا المتغير في ملف profile.

```
export ODB

CINI=/usr/pgsql/ggs/odbc.ini
```

١٤. الآن بعد أن اكتمل لدينا إعداد ODBC ، فإن الخطوة التالية هي أن تبدأ إعداد غولدنغايت ل بوستجريسكل.

سنقوم أولا بإطلاق مترجم سطر الأوامر غولدنغايت ل بوستجريسكل.

```
./ggsci
```

١٥. سنقوم الآن بإنشاء مجلدات فرعية مختلفة لتقرير غولدنغايت وملفات التعريف، وهلم جرا.

```
GGSCI > create subdirs

Creating subdirectories under current directory /usr/pgsql/ggs

Parameter files                /usr/pgsql/ggs/dirprm: already exists
```

```
Report files                    /usr/pgsql/ggs/dirrpt: created
Checkpoint files                /usr/pgsql/ggs/dirchk: created
Process status files            /usr/pgsql/ggs/dirpcs: created
SQL script files                /usr/pgsql/ggs/dirsql: created
Database definitions files      /usr/pgsql/ggs/dirdef: created
Extract data files              /usr/pgsql/ggs/dirdat: created
Temporary files                 /usr/pgsql/ggs/dirtmp: created
Stdout files                    /usr/pgsql/ggs/dirout: created
```

١٦. الخطوة التالية هي إنشاء ملف إعدادات المدير مع رقم المنفذ. هنا ندخل رقم المنفذ ٧٨٠٩ في ملف إعدادات المدير ثم قم بتشغيل المدير.

```
GGSCI > edit param mgr
GGSCI > view param mgr
PORT 7809
```

١٧. بمجرد إنشاء ملف الإعدادات يمكننا أن نبدأ المدير والتحقق من حالته.

```
GGSCI > start mgr
Manager started.

GGSCI > info all
Program     Status      Group       Lag at Chkpt  Time Since Chkpt

MANAGER     RUNNING
GGSCI > info mgr
Manager is running (IP port 7809).
```

١٨. سنقوم الآن بإنشاء جدول في كل من قاعدة بيانات أوراكل و بوستجريسكل ونسخ البيانات بين الاثنين. قم بتسجيل الدخول إلى قاعدة بيانات أوراكل وإنشاء الجدول.

```
sqlplus nkumar
SQL> create table abcd(col1 number,col2 varchar2(50));
Table created.
SQL> alter table abcd add primary key(col1);
Table altered.
```

١٩ - والخطوة التالية هي تسجيل الدخول إلى قاعدة بيانات بوستجريسكل وإنشاء جدول مماثل.

```
psql -U nkumar -d test -h dbtest
test=> create table "public"."abcd" ( "col1" integer NOT NULL, "col2"
varchar(20),CONSTRAINT "PK_Col111" PRIMARY KEY ("col1"));
```

٢٠. الخطوة التالية ستكون تسجيل الدخول إلى قاعدة بيانات أوراكل باستخدام واجهة سطر الأوامر غولدنغايت، وعرض الجداول والتقاط والتحقق من أنواع البيانات الخاصة بهم.

```
GGSCI > dblogin userid nkumar, password nkumar
Successfully logged into database.

GGSCI > list tables *
NKUMAR.ABCD

Found 1 tables matching list criteria.

GGSCI > capture tabledef nkumar.abcd
Table definitions for NKUMAR.ABCD:
COL1                          NUMBER NOT NULL PK
COL2                          VARCHAR (50)
```

٢١. في الخطوة التالية سنتحقق من اتصال ODBC لدينا إلى قاعدة بيانات بوستجريسكل واستخدام غولدنغايت CLI (واجهة سطر الأوامر) لإدراج الجداول والتقاط تعريفات الجدول.

```
GGSCI > dblogin sourcedb gg_postgres userid nkumar

Password:

2014-11-04 17:56:35  INFO    OGG-03036  Database character set identified as
UTF-8. Locale: en_US.

2014-11-04 17:56:35 INFO    OGG-03037  Session character set identified as UTF-
8.
Successfully logged into database.

GGSCI > list tables *
public.abcd
Found  1 table matching list criteria

GGSCI > capture tabledef "public"."abcd"
Table definitions for public.abcd:
col1       NUMBER (10) NOT NULL PK
col2      VARCHAR (20)
```

٢٢. في الخطوة التالية سنبدأ عملية استخراج غولدنغايت على قاعدة بيانات أوراكل. أولا سنقوم بإنشاء عملية استخراج الذي يلتقط التغييرات لجدول ABCD في قاعدة بيانات أوراكل ونسخ هذه التغييرات مباشرة إلى الجهاز بوستجريسكل. كل عملية تحتاج إلى ملف التضبيط، لذلك سنقوم بإنشاء واحد لعملية استخراج.

```
GGSCI > edit param epos
```

يتم عرض المعلمات المنشأة أدناه عند عرض ملف الإعدادات، كما يلي:

```
GGSCI > view param epos
EXTRACT epos
SETENV (NLS_LANG="AMERICAN_AMERICA.ZHS16GBK")
SETENV (ORACLE_HOME="/home/abcd/oracle/product/11.2.0/dbhome_1")
SETENV (ORACLE_SID="orapd")
USERID nkumar, PASSWORD nkumar
RMTHOST dbtest, MGRPORT 7809
RMTTRAIL /usr/pgsql/ggs/dirdat/ep
TABLE nkumar.abcd;
```

٢٣. تسمى عملية الاستخراج بـ epos وتتصل كمستخدم nkumar باستخدام كلمة المرور nkumar إلى قاعدة بيانات أوراكل. إن التغييرات التي تحدث على جدول أوراكل ABCD ستستخرج وستوضع هذه المعلومات في ملف التعقب في الجهاز بوستجريسكل. الآن بعد إنشاء ملف الإعدادات، يمكننا إضافة عملية استخراج وبدء تشغيلها.

```
GGSCI > add extract epos, tranlog, begin now
EXTRACT added.


GGSCI > add exttrail /usr/pgsql/ggs/dirdat/ep, extract epos, megabytes 5
EXTTRAIL added.

GGSCI > start epos

Sending START request to MANAGER ...
EXTRACT EPOS starting

GGSCI > info all

Program     Status      Group       Lag at Chkpt  Time Since Chkpt

MANAGER     RUNNING
EXTRACT     RUNNING     EPOS        00:00:00      00:00:00

GGSCI > info extract epos
```

٢٤. وبما أننا نكرر البيانات في بيئة غير متجانسة، حيث يحدث نسخ البيانات من أوراكل إلى بوستجريسكل، فإن العملية التي تقوم بالتحميل في بوستجريسكل تحتاج إلى تقديم المزيد من التفاصيل حول البيانات في ملف الاستخراج. ويحدث ذلك عن طريق إنشاء ملف تعريف باستخدام أداة defgen.

```
GGSCI > view param defgen
DEFSFILE /home/abcd/oracle/ggs/dirdef/ABCD.def
USERID nkumar, password nkumar
TABLE NKUMAR.ABCD;
```

٢٥. يمكننا الآن الخروج من واجهة غولدنغايت واستدعاء الأداة defgen من سطر الأوامر لإنشاء ملف تعريف وإضافة المرجع إلى ملف الإعدادات defgen.

```
./defgen paramfile ./dirprm/defgen.prm
Definitions generated for 1 table in /home/abcd/oracle/ggs/dirdef/
ABCD.def
```

٢٦. الخطوة التالية ستكون نسخ ملف defgen إلى الجهاز حيث توجد قاعدة بيانات بوستجريسكل.

```
cd /home/abcd /oracle/ggs/dirdef
scp dirdef/ABCD.def postgres@dbtest:/usr/pgsql/ggs/dirdef
```

٢٧. الخطوة التالية ستكون بدء عملية النسخ المتماثل لبوستجريسكل، وسنقوم بإعداد ملف الإعدادات لها، وتشمل ملف التعريف الذي نسخ من الخادم المستضيف لقاعدة بيانات أوراكل إلى خادم استضافة بوستجريسكل.

```
GGSCI > edit param rpos
```

المعلمات التي أنشئت عند عرض ملف الإعدادات كما هو موضح أدناه:

```
GGSCI > view param rpos
REPLICAT rpos
SOURCEDEFS /usr/pgsql/ggs/dirdef/ABCD.def
SETENV ( PGCLIENTENCODING = "UTF8" )
SETENV (ODBCINI="/usr/pgsql/ggs/odbc.ini" )
SETENV (NLS_LANG="AMERICAN_AMERICA.AL32UTF8")
TARGETDB GG_Postgres, USERID nkumar, PASSWORD nkumar
DISCARDFILE /usr/pgsql/ggs/dirrpt/diskg.dsc, purge
MAP NKUMAR.ABCD, TARGET public.abcd, COLMAP (COL1=col1,COL2=col2);
```

٢٨. في الخطوة التالية نقوم بإنشاء عملية النسخ المتماثل، وبدء تشغيلها والتحقق مما إذا كانت قيد التشغيل.

```
GGSCI > add replicat rpos, NODBCHECKPOINT, exttrail /usr/pgsql/ggs/dirdat/ep
REPLICAT added.


GGSCI > start rpos

Sending START request to MANAGER ...
```

```
REPLICAT RPOS starting

GGSCI > info all

Program      Status      Group       Lag at Chkpt  Time Since Chkpt

MANAGER      RUNNING
REPLICAT     RUNNING     RPOS        00:00:00      00:00:00


GGSCI > info all

Program      Status      Group       Lag at Chkpt  Time Since Chkpt

MANAGER      RUNNING
REPLICAT     RUNNING     RPOS        00:00:00      00:00:02
GGSCI > view report rpos
```

٢٩. الآن بعد أن أنشأت عمليات استخراج extract ونسخ replicat في واجهات غولدنغايت لـ أوراكل و بوستجريسكل، ستكون الخطوة التالية هي اختبار التضبيط. نبدأ أولا بتسجيل الدخول إلى قاعدة بيانات أوراكل وإدراج سجلات في جدول ABCD.

```
sqlplus nkumar
SQL> insert into abcd values(101,'Neeraj Kumar');
1 row created.
SQL> commit;
Commit complete.
SQL> select * from abcd;
col1 | col2
-----+-------------------
101  | Neeraj Kumar
```

٣٠. الآن سوف نتحقق إذا كانت التغييرات المقابلة / سجلات جديدة أدرجت في جدول ABCD في قاعدة بيانات أوراكل مرئية في الجدول ABCD المقابلة في قاعدة بيانات بوستجريسكل.

```
psql -U nkumar -d test
test=> select * from abcd;
 col1 |       col2
------+-------------------
  101 |  Neeraj Kumar
(1 row)
```

هذا الإعداد يكمل سيناريو اختبار لنسخ البيانات / التغييرات غير المتجانسة بين أوراكل إلى بوستجريسكل.

كيف تعمل...

بالنسبة للخطوات المذكورة في القسم السابق، سنناقش الخطوات من ١ إلى ٢٧ من القسم السابق في مقاطع.

- سوف نتحدث أولا عن الخطوات من ١ إلى ٦ من القسم السابق: في البداية نقوم بعمل اتصال ذو صلاحيات مطلقة في أوراكل مع امتياز sysdba وإجراء تغييرات معينة في التضبيط. نحن أولا نقوم بتفعيل وجهة لحفظ السجلات المؤرشفة، وهي السجلات التي تحتوي على معلومات حول تغييرات المعاملات سيحتفظ بها في الموقع المحدد من قبل إعدادات تضبيط log_archive_dest_1، كما هو موضح في الخطوة ٢ من القسم السابق. ثم نقوم بإغلاق قاعدة البيانات من أجل ضمان أن التغييرات التي أدخلت في الخطوة ٢ لتظهر في حيز التنفيذ. بمجرد إعادة تشغيل قاعدة البيانات، نقوم بتضبيط الأرشفة في قاعدة البيانات وتفعيل تسجيل تكميلي، كما هو موضح في الخطوة ٤ و ٥ من القسم السابق. في الخطوة ٦ نقوم بتضبيط تضمين مسار الدليل غولدنغايت ومسار المكتبة في PATH و LD_LIBRARY_PATH في متغيرات البيئة.
- سوف نتحدث الآن عن الخطوتين ٧ و ٨ من القسم السابق: بعد تثبيت غولدنغايت على الخادم الذي يستضيف قاعدة بيانات أوراكل، سنقوم بعد ذلك بتشغيل غولدنغايت ثم نقوم بإنشاء العديد من مجلدات غولدنغايت الفرعية لحفظ ملفات الإعدادات وملفات نقطة التفتيش وملفات تعريف قاعدة البيانات ، استخراج ملفات البيانات، وهلم جرا.
- سنتحدث الآن عن الخطوتين ٩ و ١٠ من القسم السابق: يقوم مدير غولدنغايت بعدد من الوظائف مثل بدء عملية غولدنغايت وإدارة ملف سجل التعقب وإعداد التقارير. تحتاج عملية المدير إلى التضبيط على كل من المصدر و الأنظمة المستهدفة، وتضبط بمساعدة ملف الإعدادات كما هو مبين في الخطوة ٩. نقوم بضبط المعرف PORT لتعريف المنفذ الذي يعمل عليه المدير. بمجرد أن يعد ملف الإعدادات للمدير على الجهاز المصدر، نشغل المدير ونتحقق ما إذا كان قيد التشغيل. ويظهر ذلك في الخطوة ١٠ من القسم السابق.
- سوف نتحدث الآن عن الخطوات من ١١ إلى ١٥ من المقطع السابق: بمجرد تثبيت غولدنغايت على الجهاز حيث يستضاف فيه خادم بوستجريسكل، نقوم بإضافة مجلد غولدنغايت ومسار المكتبة إلى متغيرات البيئة: PATH و LD_LIBRARY_PATH. يستخدم غولدنغايت أساسا اتصال ODBC للاتصال بقاعدة بيانات بوستجريسكل. لهذا

الغرض قمنا بإعداد ملف تضبيط ODBC يسمى odbc.ini الذي يحتوي على معلومات الاتصال للاتصال بخادم بوستجريسكل. يظهر هذا في الخطوة ١٢. في الخطوة التالية، نقوم بتصدير متغير بيئة ODBCINI وتضمين مسار ملف الإعداد. ثم من الخطوة ١٤ فصاعدا نحن نطلق واجهة سطر الأوامر غولدنغايت ل بوستجريسكل ثم نقوم بإنشاء أدلة فرعية مختلفة لحفظ ملفات الإعدادات وملفات تعريف قاعدة البيانات، وهلم جرا.

- سنتحدث الآن عن الخطوتين ١٦ و ١٧ من القسم السابق: على غرار ما نفذ في الخطوتين ٩ و ١٠ على نظام المصدر لعملية المدير في غولدنغايت لأوراكل، بطريقة مشابهة نقوم بتضبيط ملف الإعدادات للمدير عملية في غولدنغايت للنظام الهدف، وهي بوستجريسكل ومن ثم بدء تشغيل مدير ثم التحقق من ذلك، كما هو مبين في الخطوة ١٧. المعلمة الوحيدة التي تم أعدت في الخطوة ١٦ هي معلمة المنفذ PORT التي حددت المنفذ الذي يستمع المدير إليه.
- سنتحدث الآن عن الخطوتين ١٨ و ١٩ في القسم السابق: نحن هنا نضع جدولين بنفس الأسماء، لهما نفس البنية. وسينشأ جدول واحد في قاعدة بيانات أوراكل و واحد في بوستجريسكل. تنشأ الجداول بهذه الطريقة لأن فكرة هذه العملية هي أن أي تغييرات في البيانات التي تحدث على الجدول المنشأ في أوراكل سينسخ / ينشر في بوستجريسكل. هذا هو مفهوم النسخ المتماثل غير المتجانس.
- هنا سوف نتحدث عن الخطوات ٢٠ و ٢١ من القسم السابق: في الأساس، في الخطوة ٢٠ ما نقوم به هو تسجيل الدخول إلى قاعدة بيانات أوراكل باستخدام واجهة غولدنغايت و نلتقط تعريف الجدول للجدول الذي أنشأ في الخطوة ١٨ من القسم السابق. وبالمثل، في الخطوة ٢١، نحن نتحقق من اتصال ODBC إلى قاعدة بيانات بوستجريسكل من واجهة غولدنغايت وبمجرد إجراء اتصال نحن نلتقط تعريفات الجدول للجدول الذي أنشأ في الخطوة ١٩.
- هنا سنتحدث عن الخطوات ٢٢ و ٢٣ من القسم السابق: في الخطوة ٢٢ نقوم بإنشاء ملف معلمات لعملية الاستخراج extract على الجهاز الذي يستضيف قاعدة بيانات أوراكل لأنه سيستخدم كمصدر. عملية الاستخراج تحدث على قاعدة بيانات المصدر. يحتوي ملف إعدادات عملية الاستخراج على معلومات تتعلق ببيئة أوراكل والمضيف البعيد المستهدف ومنفذ المدير وملف التعقب والجدول الذي يجب تسجيل التغييرات عليه. في الخطوة ٢٣، نبدأ عملية الاستخراج على قاعدة بيانات أوراكل المصدر ونضيف ملف التعقب. ستستخرج عملية extract أية تغييرات على جدول أوراكل ABCD وستضع هذه المعلومات على ملف التعقب الذي يقيم على الجهاز استضافة خادم بوستجريسكل.
- نحن هنا سنتحدث عن الخطوات ٢٤ و ٢٥ من القسم السابق: كما يحدث النسخ المتماثل في بيئة غير متجانسة، وهنا من أوراكل إلى بوستجريسكل في هذا السيناريو، فمن المهم الحصول على أكبر قدر ممكن من التفاصيل

حول البيانات في ملف الاستخراج لجعل الأمور واضحة لعملية تحميل البيانات إلى قاعدة بيانات بوستجريسكل. ولكي يحدث ذلك، نحتاج إلى إنشاء ملف تعريف على واجهة غولدنغايت لقاعدة بيانات أوراكل ثم يشحن إلى الجهاز الذي يستضيف خادم بوستجريسكل في الخطوة ٢٤، ونحن أساسا ننشئ ملف الإعدادات للأداة defgen. في الخطوة ٢٥، نستدعي الأداة defgen لإنشاء ملف تعريف ونضيف مرجعا إلى ملف الإعدادات الذي أنشأ في الخطوة ٢٤ من القسم السابق.

- في الخطوة ٢٦ من القسم السابق، نقوم بنسخ ملف التعريف الذي أنشأ في الخطوة ٢٥ من الجهاز أوراكل إلى الجهاز الذي يستضيف خادم بوستجريسكل.
- هنا سنتحدث عن الخطوتين ٢٧ و ٢٨ من القسم السابق. هنا نبدأ عملية النسخ المتماثل. عملية النسخ المتماثل تقرأ أساسا التغييرات من ملف التعقب وتوزعها على قاعدة بيانات بوستجريسكل. في الخطوة ٢٧ نقوم ببساطة بضبط ملف الإعدادات لعملية النسخ المتماثل وفي الخطوة ٢٨ نبدأ عملية النسخ المتماثل replicat ونضيف ملف التعقب إلى عملية النسخ المتماثل بحيث يمكن قراءة التغييرات من ملف التعقب وتفريغ تلك التغييرات إلى بوستجريسكل قاعدة البيانات.
- سوف نتحدث هنا عن الخطوات ٢٩ و ٣٠. في الأساس نقوم باختبار التضبيط لدينا هنا. في الخطوة ٢٩، نقوم بتسجيل الدخول إلى قاعدة بيانات أوراكل، وإدراج سجل في جدول ABCD وحفظ التغييرات. الآن مع عمل عمليتنا استخراج غولدنغايت وعملية النسخ المتماثل فإن السجلات التي أدرجت حديثا في جدول أوراكل يجب أن تكون طبقت على الجدول المقابل في قاعدة بيانات بوستجريسكل. نؤكد هذا عن طريق تسجيل الدخول إلى قاعدة بيانات بوستجريسكل ثم تحديد السجلات من جدول ABCD في الخطوة ٣٠. يمكننا أن نرى في الخطوة ٣٠ أن السجلات المدرجة في الخطوة ٢٩ في جدول أوراكل مرئية في جدول قاعدة بيانات بوستجريسكل ABCD. وهذا يؤكد نجاح تنفيذ النسخ المتماثل غير المتجانسة من قاعدة بيانات أوراكل إلى قاعدة بيانات بوستجريسكل.

تم الكتاب بحمد الله